



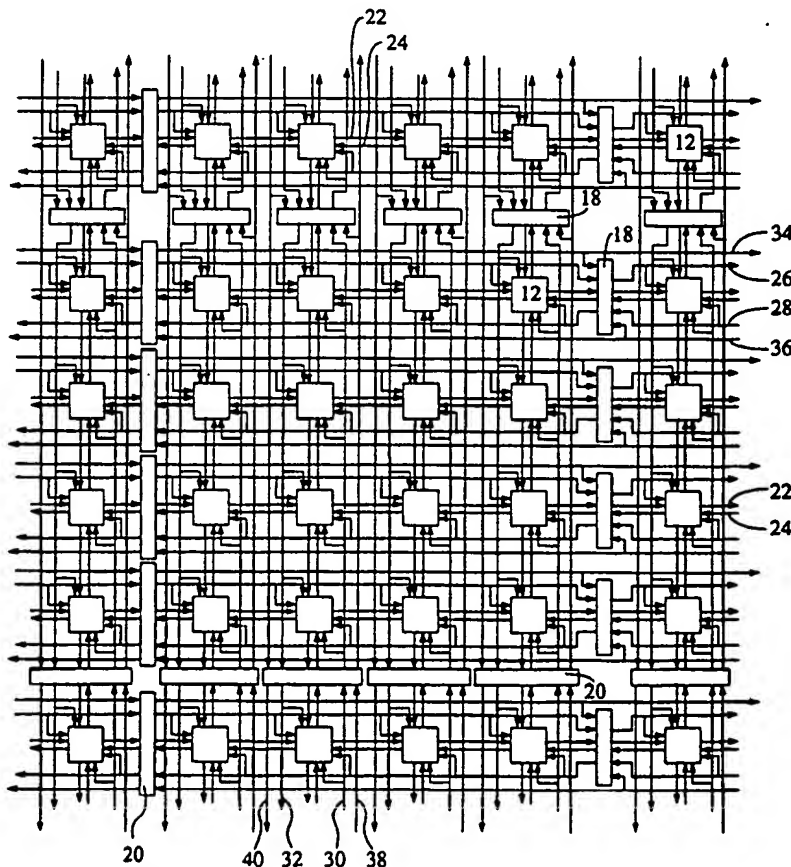
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁵ : H03K 19/177	A1	(11) International Publication Number: WO 94/10754 (43) International Publication Date: 11 May 1994 (11.05.94)
---	-----------	---

(21) International Application Number: **PCT/US93/10404**(22) International Filing Date: **5 November 1993 (05.11.93)**(30) Priority data:
9223226.3 **5 November 1992 (05.11.92)** **GB**(71) Applicant: **XILINX, INC. [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US).**(72) Inventor: **KEAN, Thomas, A. ; 1/6 Gilmours Entry, Edinburgh EH8 9XL (GB).**(74) Agents: **YOUNG, Edel, M. et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).**(81) Designated States: **CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).****Published**
With international search report.(54) Title: **IMPROVED CONFIGURABLE CELLULAR ARRAY**

(57) Abstract

A field programmable gate array (FPGA) of cells (12) arranged in rows and columns is interconnected by a hierarchical routing structure. Switches (18, 20) separate the cells (12) into blocks and blocks of blocks with routing lines (26, 28, 30, 32, 34, 36, 38, 40) interconnecting the switches (18, 20) to form the hierarchy.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

IMPROVED CONFIGURABLE CELLULAR ARRAY3 FIELD OF THE INVENTION

4 The present invention relates to a configurable
5 cellular array of dynamically configurable logic elements,
6 such arrays being generally known as Field Programmable Gate
7 Arrays (FPGAs).

9 BACKGROUND OF THE INVENTION

10 Reprogrammable FPGAs have been available commercially
11 for several years. The best known commercial family of
12 FPGAs are those from Xilinx Inc. One class of these devices
13 uses Static Random Access Memory (SRAM) to hold control bits
14 which control their configurations. Most FPGA devices
15 replace traditional mask programmed Applications Specific
16 Integrated Circuit (ASIC) parts which have a fixed
17 configuration. The configuration of the FPGA is static and
18 is loaded from a non-volatile memory when power is applied
19 to the system. Nearly all commercially available FPGAs have
20 a stream-based interface to the control store. (The control
21 store contains the set of bits which determine what
22 functions the FPGA will implement.) In a stream-based
23 interface to the control store, a sequence of data is
24 applied to a port in the FPGA to provide a complete
25 configuration for the whole device or for a fixed (normally
26 large) sub-section of the FPGA. This stream-based
27 interface, when combined with an address counter which is
28 implemented on the FPGA itself, provides an efficient method
29 of loading the complete device configuration from adjacent
30 EPROM or other non-volatile memory and power up without any
31 additional overhead circuits. A stream based interface with
32 an address counter is a suitable programming interface for
33 an FPGA which is used as a replacement for a standard ASIC.
34 Some FPGAs can be partly or totally reconfigured using one
35 of a set of static configurations stored at different
36 addresses in an EPROM, and can trigger the reconfiguration
37 from within the design implemented on the FPGA.

1 Published International Application WO 90/11648,
2 corresponding to U.S. Patent 5,243,238, discloses an
3 architecture hereafter referred to as CAL I, which has been
4 implemented in an Algotronix product designated CAL 1024.
5 CAL I is different from other commercially available FPGAs
6 in that its control store appears as a standard SRAM to the
7 systems designer, and can be accessed using address bus,
8 data bus, chip enable, chip select and read/write signals.
9 Addressing the control store as an SRAM supports a user
10 program running on the host processor mapping the FPGA
11 control store (configuration memory) into the memory or
12 address space of the host processor so that the processor
13 can configure the FPGA to implement a user-defined circuit.
14 This arrangement, which is implemented in the CAL 1024 FPGA,
15 allows the user to partition an application between the
16 processor and the FPGA with appropriate sections being
17 implemented by each device. The control store interface
18 provides an important input/output (I/O) channel between the
19 FPGA and the processor, although the I/O can also take place
20 using more traditional techniques via, for example, a shared
21 data memory area. This latter type of FPGA provides a
22 passive control store interface because an external agent is
23 required to initiate configuration or reconfiguration of the
24 device, as required.

25 Experience with the CAL I architecture and trends
26 within the electronics industry have made this second
27 passive form of control store interface increasingly
28 attractive for many applications. Microprocessors or
29 microcontrollers are now pervasive components of computer
30 systems and most board level systems contain one. The major
31 benefit of the stream based "active" FPGA programming
32 approach is that no overhead circuits are required to
33 initiate reconfiguration. In systems where a microprocessor
34 or microcontroller is present, the "passive" RAM emulating
35 FPGA interface is preferable for several reasons:

36 (1) the FPGA configuration can be stored in the
37 microprocessor's program and data memory (reducing the

1 number of parts by removing the need for a separate
2 memory chip),

3 (2) the existing data and address buses on the board
4 can be used to control the FPGA (saving printed circuit
5 board area by removing dedicated wires between the
6 configuration EPROM and the FPGA);

7 (3) the FPGA control store can be written to and read
8 from by the microprocessor, and thereby used as an I/O
9 channel between the FPGA and the microprocessor,
10 thereby potentially saving additional wiring between
11 the FPGA and the processor buses and freeing the FPGA
12 programmable I/O pins for communication with external
13 devices, and

14 (4) the intelligence of the microprocessor can be used
15 to support compression schemes for the configuration
16 data and other techniques, which allows more
17 flexibility in reprogramming the FPGA.

18 In addition, the difference in cost between an "active" FPGA
19 with an associated EPROM holding its configuration and a
20 passive FPGA with an active microcontroller chip containing
21 an EPROM and a simple processor is minimal. The easy
22 reprogrammability makes the passive FPGA attractive, even if
23 the microcontroller has no other function apart from
24 reprogramming the FPGA.

25 Another trend within the Electronics Industry has been
26 the provision of "support chips" for microprocessors which
27 provide an interface between I/O devices and a particular
28 microprocessor. Examples of these devices include Universal
29 Asynchronous Receiver Transmitters (UARTs) for low bandwidth
30 serial I/O, Programmable Peripheral Interfaces (PPIs) for
31 low bandwidth parallel I/O and various specialised chips for
32 higher bandwidth connections to networks and disk drives.
33 These support chips appear to the processor as locations in
34 the I/O or memory address space to and from which data are
35 transferred. Some support chips can interrupt the processor
36 via interrupt lines or take over the bus for Direct Memory
37 Access (DMA) operations. In many ways a passive FPGA chip

1 can be viewed as a successor to a support chip, providing an
2 interface to the processor via its control store on the one
3 hand, and an interface to the external world via a large
4 number of flexible I/O lines on the other, for example 128
5 programmable I/O lines on the Algotronix CAL 1024 device.

6 A passive FPGA chip has a number of advantages. For
7 example, it is cost-effective to provide a single FPGA with
8 a library of configurations instead of providing a number of
9 support chips. In addition, providing a single FPGA for
10 several functions reduces the number of devices in the
11 processor manufacturer's catalogue. Also, reconfigurable
12 FPGAs can support changeable I/O functions, such as when a
13 single external connector can be used as either a serial or
14 a parallel port. With a passive RAM control interface, the
15 FPGA is able to support other functions as well.

16 Each time an FPGA is reconfigured to implement a
17 different set of functions, the microprocessor must access
18 the configuration memory. One reconfiguration typically
19 requires many control store accesses, one access for each
20 word of configuration memory to be changed. Several
21 important classes of reconfiguration have been identified.

22 (1) Application swapping occurs when one application
23 terminates and a completely different application
24 wishes to make use of the FPGA. In this case the FPGA
25 chip is completely reconfigured, usually from a static
26 configuration.

27 (2) Task swapping occurs when the application must
28 configure relatively large sections of the FPGA to
29 implement a new phase in the computation. For example,
30 a sorting application might first sort small batches of
31 data completely using configuration A and then merge
32 those sorts into a completely sorted stream of data
33 using configuration B. In this case, the application
34 has knowledge of both configurations and need only
35 change those resources which are different in
36 configuration B. At a later point, configuration A may
37 itself be restored.

1 (3) Data dependent reconfiguration occurs when the
2 configurations of some cells are computed dynamically
3 based on input data by the application program, rather
4 than being loaded from a static configuration file.
5 Often a static configuration is first loaded, then a
6 relatively small sub-set of cells are reconfigured
7 dynamically (that is, reconfigured while the chip is
8 operating). An important example of this class of
9 reconfiguration is where an operand (such as a constant
10 multiplier or a search string) is folded directly into
11 the logic used to implement the multiply or sort unit
12 rather than being stored in a register. This technique
13 is advantageous as it frequently results in smaller and
14 faster operation units.

15 (4) Access to gate outputs occurs for debugging. The
16 outputs of all the logic cells on the CAL I FPGA are
17 mapped to bits of the control store. Debugging
18 programs are available which read back this information
19 on the display or design layout to show the logic
20 levels on internal wires.

21 (5) Access to gate outputs for I/O is similar to the
22 previous access to gate outputs for debugging. But in
23 this particular case only a small fraction of the logic
24 nodes, namely those which correspond to input and
25 output registers, will be accessed repeatedly. The
26 ability to rapidly assemble a word representing input
27 to or the result of a computation from several bits at
28 different locations in the control store is critical to
29 the effectiveness of this technique.

30 It is desirable to reduce the number of accesses
31 required and hence the time to wholly or partially
32 reconfigure the device. Several systems other than CAL I
33 have been proposed which allow direct access to internal
34 signals in an FPGA or an FPGA-like device, for example, as
35 disclosed in Cellular Logic-in-Memory Arrays, William H.
36 Kautz, IEEE Transactions on Computers Vol C18 No. 8, August
37 1969; A Logic in Memory Computer, Harold S. Stone, IEEE

1 Transactions on Computers, Vol C19 No. 1, January 1970 and
2 Xilinx U.S. Patent No. 4,758,985 Microprocessor Oriented
3 Configurable Logic Element, although all these proposals
4 suffered from major drawbacks and were not made available
5 commercially.

6 It is also desirable to improve the means of accessing
7 state information in designs implemented on FPGAs so that an
8 external processor can perform word-wide read or write
9 operations on the registers of the user's design with a
10 single access to the control store. Thus the control store
11 interface allows high bandwidth communication between the
12 processor and the FPGA. It is also desirable to provide
13 mechanisms for synchronising computations between the FPGA
14 and the processor and to provide a mechanism for extending
15 design configuration files to support dynamic
16 reconfiguration while allowing use of conventional tools for
17 static designs to create FPGA configurations.

18 The architecture of the CAL 1024 was based on 1.5
19 micrometre technology available in 1989. One problem with
20 the CAL I architecture in which cells are connected only to
21 their nearest neighbours was that cells in the middle of the
22 array became less useful with increasing array size as the
23 distance and hence delay to the edge of the chip increased.
24 This problem became more serious as improvements in
25 processing technology meant that the number of cells
26 implementable per chip increased from 1024 to about 16,384.
27 This resulted in a scalability problem because of increased
28 delays, and reduced the performance below the desired
29 criteria. Thus, although scalability of chips using the CAL
30 I architecture can be achieved, it is at the expense of
31 performance. The limited number of cells available on a
32 single chip with 1.5 um technology meant that it was
33 desirable to ensure scalability over chip boundaries so that
34 large designs typical of many computational applications
35 could be realised using multiple chips. The limitations of
36 the then processing technology also made it essential to
37 optimise the architecture for silicon area and sometimes

1 this optimisation was at the expense of speed. The original
2 Algotronix CAL 1024 chips were designed to bring out
3 peripheral array signals to pads on the edges of the
4 cellular array so that they could be cascaded into larger
5 cellular arrays on a printed circuit board. Packaging
6 technology has not evolved as rapidly as chip technology and
7 limitations on the number of package I/O pins make it
8 uneconomic to produce fully cascadable versions of the
9 higher cell density chips.

10 The CAL I architecture suffered from a number of other
11 disadvantages. For example, in order to access a cell in
12 the existing CAL I FPGA, five to six processor instructions
13 are needed to calculate the address of the cell; this again
14 takes time and slows operation. With the existing CAL I
15 cell array the routing architecture used meant that with
16 increased number of cells per chip, routing via intermediate
17 cells added considerably to the delays involved. In
18 addition, in the CAL 1024 device, global signals are coupled
19 to all the cells in the array so that the cells can be
20 signalled simultaneously. It logically follows that at high
21 clock frequencies, global signals could consume high power.

22

23 SUMMARY OF THE INVENTION

24 An object of the present invention is to provide an
25 improved field programmable gate array which obviates or
26 mitigates at least one of the aforementioned disadvantages.

27 A further object of the present invention is to reduce
28 the number of control store accesses required and the time
29 to wholly or partially reconfigure the device from one
30 configuration to another.

31 A further object of the invention is to enable an
32 external processor to perform word-wide read or write
33 operations on registers of a user's design with a single
34 access to the control store.

35 A yet further object of the present invention is to
36 provide a mechanism for extending design configuration files
37 to support dynamic reconfiguration while allowing the use of

1 conventional tools for static designs to create FPGA
2 configurations.

3 A further object of the invention is to provide
4 mechanisms for the synchronisation of computations between
5 the FPGA and an external processor.

6 A yet further object of the invention is to provide a
7 novel routing architecture which can be scaled up to operate
8 on arrays having different numbers of cells to reduce delays
9 involved in routing between cells in large cell arrays.

10

11 Array of Cells with Hierarchical Routing Structure

12 In accordance with the present invention, a 2-
13 dimensional field programmable gate array (FPGA) of cells
14 arranged in rows and columns is provided. Each cell in the
15 array has at least one input and one output connection at
16 least one bit wide to each of its neighbouring cells. Each
17 cell also has a programmable routing unit and a programmable
18 function unit to permit intercellular connections to be
19 made. The programmable function unit can select one of a
20 plurality of functions of several input signals for
21 generating a function unit output signal. The routing unit
22 of a cell directs inputs of the cell to function unit
23 inputs, and also directs inputs of the cell and the function
24 unit output to neighbouring cells. Groups of cells in the
25 array are arranged so as to be connected to additional
26 conductors of a length equal to a predetermined number of
27 cells. Cells in the array are coupled to the additional
28 conductors via switches. Typically, four such conductors
29 are provided for each cell, two conductors arranged in one
30 direction in the array and two conductors arranged in the
31 orthogonal direction in the array. Each pair of conductors
32 is arranged such that one conductor in the pair carries
33 signals in one direction and the other conductor carries
34 signals the opposite direction. This novel architecture is
35 referred to hereafter as the CAL II architecture, or simply
36 as CAL II.

37 A predetermined block of cells, for example a 4 x 4

1 block of cells, has the additional conductors of at least
2 cell length 4 (four cells long). These blocks are arranged
3 into repeating units to form an array of these cells whereby
4 16 of such 4 x 4 blocks of cells result in a unit of 16
5 cells x 16 cells with each 4 x 4 block having the additional
6 conductors, the longer conductors hereinafter referred to as
7 flyovers, associated with each row or column of 4 cells.
8 The 16 x 16 block of cells may itself have additional
9 flyover conductors.

10 In larger cellular arrays, the structure of the
11 hierarchical routing can be extended to any number of
12 levels, a third level using conductors of length 64, and a
13 fourth level using conductors of length 256 and so on.

14 This arrangement permits scaling of the array with the
15 advantage that the scaling is logarithmic in terms of
16 distance, thereby significantly reducing delay between
17 cells. Specifically, a signal travels from an origin cell
18 to its closest associated switch located at a cell block
19 boundary, then along appropriate flyovers to the destination
20 cell. Thus, this structure creates a hierarchical cellular
21 array with a variety of routing resources whose lengths
22 scale, in one embodiment by a factor of 4 each time, built
23 on top of a simple array of neighbour connections only.

24 The principal advantage of providing different levels
25 of routing resources is that it allows the number of
26 conductor segments required to route from cell to cell on
27 the array to be minimised. For example, if a path is
28 provided between two points in the array using neighbour
29 interconnect only, the number of routing segments would be
30 equal to the number of cells between the two points, whereas
31 with the hierarchical interconnect, the number of segments
32 increases with the logarithm of the distance between the
33 cells.

34

35 Single-Source Directed Wiring

36 In one embodiment of a pre-programmable cellular array
37 with hierarchical routing, all the wires in the array are

1 directed and have a single source. Thus, 3-state drivers
2 are not used. In one embodiment, all the connections in the
3 array are completely symmetrical so that if the array is
4 rotated the structure remains unchanged. Single source
5 wiring has the advantage of being simpler to implement than
6 multiple-source wires. Multiple-source wires, while
7 allowing additional flexibility, require a considerable area
8 overhead, and produce contention when different drivers
9 attempt to place opposite values on the same wire.
10 Contention dissipates heavy power, which can result in
11 device failure. Contention is obviated by the present
12 invention, in which a wire is driven by a single multiplexer
13 output. The symmetry feature simplifies the CAD software to
14 map user designs onto the array and allows hierarchical
15 blocks of those designs to be rotated or reflected to
16 provide better utilisation of the array area.

17 Preferably, the switches providing connections between
18 the flyovers and the cells are static RAM controlled
19 multiplexers. Conveniently, the switches at 4-cell and 16-
20 cell boundaries permit direct neighbour connections as well
21 as additional connections via the longer flyover conductors.
22

23 Automatic Routing Optimization, Portability

24 The hierarchical routing resources in the improved FPGA
25 can be used in two principal ways. Firstly, the user can
26 design using simpler neighbour programming models ignoring
27 the availability of the longer connections. In such a case,
28 the software will automatically detect nets in the layout
29 which may benefit from new routing resources, and take
30 advantage of these resources to speed up performance of the
31 design. Secondly, the user can design using improved
32 programming models and make explicit assignments to the
33 extra routing resources. In this case, extra density is
34 achieved by assigning different nets to various levels of
35 interconnect at the same point in the cellular array. For
36 example, a length-16 wire could carry a signal over a sub-
37 unit (for example, several 4 x 4 blocks) without interfering.

1 with the local interconnect in that sub-unit. When flyovers
2 are used to bypass a block of cells, blocks of the user
3 design might have to be placed in the FPGA on these 4-cell
4 or 16-cell boundaries. Automatic addition of flyover
5 routing is easier to use and is independent of the number of
6 levels of routing provided by a given FPGA chip. Using
7 software to add the flyovers provides design portability
8 between different chips, and using improved programming
9 models which use flyovers to bypass a block, or manually
10 assigning flyover resources as appropriate, allows more
11 efficient use of the resources provided.

12 Use of longer routing resources may be achieved using
13 low level CAD software as described above or using hardware
14 in the chip itself to automatically route signals to longer
15 wires where possible. This provides more device portability
16 and allows special "fast" versions of existing chips to be
17 made with additional longer wires without requiring any
18 changes to the existing design. This "dynamic" selection of
19 longer routing wires simplifies the CAD software, allowing
20 it to run faster. Dynamic selection of longer wires is
21 particularly attractive for applications which involve
22 dynamically reprogramming FPGA chips.

23 According to another aspect of the invention the speed
24 of propagation of signals through an FPGA is improved by
25 automatically mapping onto flyovers those signals capable of
26 being speeded up using circuitry fabricated on the FPGA.
27 The method comprising the steps of:

28 detecting control store bit patterns which correspond
29 to routing a signal straight through a cell, detecting when
30 a group of cells beneath a flyover all route the signal in
31 the direction of the flyover by using the 4-input gate
32 provided for that flyover direction, and taking as input the
33 output of the 4-input gate of the appropriate neighbour
34 multiplexer,

35 feeding an output from one of the 4 input gates to
36 switches at both ends of the flyover, whereby the signal is
37 carried automatically by the flyover as well as by neighbour

1 routing, and the faster signal on the flyover is selected by
2 the switch at the end of the flyover.

3 The method is scalable and can be applied to a group of
4 4 length-4 flyovers under a length-16 flyover when this
5 group all route a signal in the direction of the length-16
6 flyover. This is done using a 4-input gate which takes as
7 inputs the outputs of the 4-input gates used for receiving
8 signals from the neighbour cells.

9 The type of gate used depends on the control bits being
10 detected. For example, a NOR gate is used for detecting
11 bits 0,0 in an East to West direction in a West routing
12 multiplexer. Alternatively, to detect a bit pattern of 1,1,
13 a NAND gate and associated logic circuitry are used.

14

15 Block Correspondence Allows Easy Reconfiguration

16 An important feature of the present invention, is that
17 a rectangular area of cells specified as a hierarchical
18 block of the user's design (for example, a 4 x 4 block or a
19 16 x 16 block of cells) corresponds directly, i.e. by a
20 straight-forward physical relationship, with one or more
21 rectangular areas in the configuration memory (control
22 store) of the CAL II FPGA device representing instances of
23 that block. This means that a block of the user's design
24 can be dynamically replaced by another block of the same
25 size, for example, a register can be replaced by a counter
26 of equal size. Thus, in accordance with the present
27 invention, the host processor must reconfigure only the
28 corresponding area of the control store RAM. The binary
29 data for both blocks can be pre-calculated from the user's
30 design, and the actual replacement can be done very rapidly
31 using a block transfer operation, as is well known in the
32 art. During dynamic reconfiguration, registers can be
33 initialised either to a default associated with a block
34 definition or to restore the previous state of the unit
35 whose configuration is being restored or to a convenient
36 value decided by the application program performing the
37 reconfiguration.

1

2 Wildcard Feature

3 According to a further aspect of the present invention
4 an FPGA is provided having a randomly accessible writable
5 control store in which more than one word of control memory
6 is written simultaneously as a result of a single write
7 access. Conveniently, the row and column decoders may be
8 implemented by standard NOR gates coupled to wildcard
9 registers associated with the address buses to the
10 respective row and column decoders.

11

12 Match Feature

13 Alternatively, the FPGA includes a plurality of
14 programmable rather than fixed row decoders, which are
15 implemented by means of match registers. Also, the FPGA
16 includes a plurality of column decoders which are
17 implemented by match registers.

18

19 Shift and Mask Feature

20 According to a further aspect of the present invention
21 shift and mask registers are provided between an external
22 data bus and internal data bus to the bit line drivers.
23 This has the advantage of allowing for additional
24 flexibility in selecting which bits of the addressed word
25 are significant for the current transfer and presenting that
26 information in a more convenient form, such as left aligned
27 to the external processor.

28 Preferably, the FPGA writable control store includes a
29 mask unit for allowing some bits of a word to be programmed
30 selectively. Conveniently, the mask unit includes shift
31 components which can expand left aligned data for unmasked
32 bits or produce left aligned data from a word with some bits
33 masked out.

34 In the FPGA, word-wide read and write accesses may be
35 made through the control store interface to registers of a
36 user design. Register access can be extended to an
37 antifuse, EPROM, EEPROM or mask programmable logic device by

1 providing an additional RAM like interface for accessing
2 internal state information.

3

4 Configuration and State Information Segregated

5 Advantageously, in the CAL II FPGA, the values present
6 on internal nodes appear in the control store address space
7 such that any word in the address space contains bits
8 representing values on internal nodes or bits containing
9 configuration information, but not both. Conveniently, the
10 values in internal nodes appear in the control store address
11 space such that addresses corresponding to state information
12 are distinguishable from addresses corresponding to
13 configuration information by examining one or a small sub-
14 set of mode bits from the address bus.

15 Conveniently, the FPGA includes a further set of bit
16 and word line drivers which are arranged orthogonally to the
17 first set of bit and word line drivers such that logic state
18 information in a dual-ported memory in the device is
19 accessible word-wide in either the horizontal (bit) or
20 vertical (word) direction.

21

22 Multiple Address Decoders

23 According to another aspect of this invention bit and
24 word lines in the RAM are associated with multiple address
25 decoders, and additional address bits are fed to these
26 secondary decoders. Using more than one address decoder
27 allows a more complex mapping between internal memory bits
28 and external addresses, including the possibility of
29 multiple bits of memory having a single address. This
30 technique allows for density of the memory array while
31 preserving logical fields corresponding to different device
32 functions in the external address.

33

34 Microcontroller Integrated with FPGA

35 A further aspect of the invention integrated FPGA
36 architecture on the same chip with a microprocessor or
37 microcontroller, wherein the FPGA control store memory is

1 mapped into the processor address space.

2

3 On-Chip Timers

4 The FPGA architecture may include programmable counter-
5 timers integrated on the chip to drive in global clock
6 signals.

7

8 External and Internal Programmability

9 The address and data buses in the CAL II FPGA used for
10 programming can also be connected to cell inputs and outputs
11 as well as external signals.

12 To external systems, the CAL II array appears as two
13 separate devices: a static random access memory control
14 store and a programmable logic unit which is effectively the
15 user's design. The memory control store is normally mapped
16 into the address space of the host computer allowing rapid
17 changes of configuration. Use of random access control
18 memory in the CAL II FPGA means that only those cells or
19 parts of cells whose function has changed need to be
20 reprogrammed. It will be understood that the programmable
21 logic unit consists of the array of functional cells
22 surrounded by programmable input/output blocks.

23

24 High Speed Path

25 According to another aspect of the present invention,
26 the function unit of a cell has a plurality of input signals
27 for receiving a plurality of input variables, the input
28 variables being processed in parallel paths, whereby one of
29 the parallel paths is optimised for speed so that a user can
30 direct critical signals to the optimised path to ensure
31 minimal delay in processing the signal.

32

33 Reconfiguration Synchronized with Computation

34 According to a further aspect of the invention there is
35 provided a method of writing data directly into a register
36 of the CAL II array. The method comprises the steps of
37 using the bit or word lines to the control store as clocks

1 or signals for synchronising computations in a circuit
2 implemented in the CAL II array. In this manner, user
3 logic implemented on the FPGA is synchronized to a
4 microprocessor.

5 According to a further aspect of the invention,
6 external signals may be monitored by circuits implemented on
7 an FPGA. The monitoring method comprises the steps of
8 connecting the external signals to be monitored to positions
9 at the periphery of the cell array which do not have
10 associated I/O pads. Available external signals include
11 data bus, address bus, mode, read/write, chip enable, chip
12 select, reset, and force high impedance (FHZ).

13 Advantageously, a circuit implemented on the FPGA
14 detects external reads and writes to the control memory, and
15 automatically clocks itself to process an input value or
16 produce the next output value.

17 According to yet a further aspect of the present
18 invention there is provided an FPGA in which circuits are
19 implementable as one or more replaceable blocks,
20 configuration data for each potential configuration of each
21 replaceable block being storable in memory, the replaceable
22 blocks being selected from a hierarchical selection of
23 blocks with associated bounding box sizes, the blocks being
24 replaceable by alternative configurations having the same
25 bounding box and I/O signals appearing at the same point on
26 the block periphery.

27 These and other aspects will become apparent from the
28 following description when taken in combination with the
29 accompanying drawings.

30

31 BRIEF DESCRIPTION OF THE DRAWINGS

32 Fig. 1 is an enlarged plan view of a representative
33 area of an FPGA in accordance with a preferred embodiment of
34 the invention, showing the spatial 2-dimensional
35 relationship between individual cells and switches, the
36 representative relevant area having 16 cells x 16 cells;

37 Fig. 2 is an enlarged portion in more detail of the

1 area indicated in broken outline of Fig. 1 showing the
2 routing paths between the cells and the switches in the
3 representative area;

4 Fig. 3 is a schematic block diagram of the
5 representative area shown in Fig. 1 at its basic routing
6 level, that is with neighbour interconnects only;

7 Fig. 4 depicts the array structure shown in Fig. 1,
8 with the neighbour interconnects omitted in the interest of
9 clarity, and with additional first level routing flyovers
10 for a 4-cell x 4-cell block;

11 Fig. 5 is a view similar to Fig. 4 but with the length-
12 4 flyovers omitted, and shows the routing flyover
13 arrangement for the entire 16-cell x 16-cell block of Fig. 1
14 and depicts length-16 flyovers only;

15 Fig. 6 is a schematic diagram showing a row of cells
16 and how a signal may be passed along the line of cells using
17 the neighbour connect and the length-4 flyover arrangement
18 shown in Figs. 3 and 4;

19 Fig. 7 is a schematic diagram of a 64 x 64 cell array
20 which embodies the CAL II architecture, showing the I/O
21 units along the perimeter of the chip;

22 Fig. 8 is a block diagram of the above device showing
23 the row and column decode units, control and global I/O
24 units and global buffers of this embodiment of the CAL II
25 architecture device;

26 Fig. 9 is a logic symbol for the CAL II array shown in
27 Fig. 8 used in schematic diagrams of systems containing the
28 above embodiment of the CAL II architecture;

29 Fig. 10 is an enlarged schematic diagram of one of the
30 cells shown in the representative area of Fig. 2;

31 Fig. 11 shows functions implementable by function unit
32 48 of the cell shown in Fig. 10;

33 Fig. 12 is another representation of output elements
34 50, 52, 54, and 56 of in Fig. 10;

35 Fig. 13 is a schematic diagram of a switch at a 4-cell
36 boundary showing the interconnections to the switch;

37 Fig. 14 is a schematic diagram of a switch at a 16-cell

1 boundary showing the interconnections to the switch;

2 Fig. 15 is a schematic layout of a switching function
3 for use at 4-cell boundaries;

4 Fig. 16 is similar to Fig. 15 and depicts an extended
5 switching function for use at 4-cell boundaries with global
6 signals and connections to programming bit signals;

7 Fig. 17 is a similar schematic diagram but for a
8 preferred switching function for use at 16-cell boundaries;

9 Fig. 18 depicts a NOR gate circuit for automatically
10 moving local signals to a flyover.

11 Fig. 19 depicts one implementation of the NOR gate of
12 Fig. 18.

13 Fig. 20 depicts a first version of a function unit in
14 accordance with the preferred embodiment of the present
15 invention;

16 Fig. 21 depicts an alternative version of a function
17 unit for use with the preferred embodiment of the present
18 invention;

19 Fig. 22 depicts a further alternative version of a
20 function unit for use with the preferred embodiment of the
21 present invention, this version reducing the symmetry of the
22 input selection multipliers;

23 Fig. 23 is a further embodiment of a function unit
24 similar to that shown in Fig. 20;

25 Fig. 24 depicts a schematic diagram of a 2-input
26 function unit for use with the present invention;

27 Fig. 25 is a schematic diagram of a 3-input function
28 unit for use with the present invention;

29 Fig. 26a depicts a preferred input/output architecture
30 arrangement for each pad used with the FPGA of the present
31 invention;

32 Fig. 26b is a table of the various I/O block modes for
33 the architecture shown in Fig. 26a;

34 Fig. 26c shows an example circuit in which a global
35 signal can be taken from several sources by a user
36 programmable multiplexer.

37 Fig. 26d shows a schematic diagram of a switch used in

1 the input/output architecture of Fig. 26a.

2 Fig. 27a shows the memory map and row and column
3 address structures for accessing the register or node output
4 of each cell in the CAL II array.

5 Fig. 27b is a diagrammatic representation of a RAM
6 programmed FPGA which consists of an array of RAM cells
7 within which are embedded active logic structures;

8 Fig. 27c is an enlarged representation of the RAM
9 programmed FPGA shown in Fig. 27b depicting the word
10 boundaries in the control store;

11 Fig. 28 depicts a RAM programmed array in which the row
12 and column decoders have been replaced by match registers;

13 Fig. 29 is a table depicting the address bus format for
14 use with FPGA implemented by an embodiment of the CAL II
15 architecture having 4096 cells;

16 Fig. 30 is a table depicting the area of the control
17 store which is selected by using particular mode values;

18 Fig. 31a is a table showing the cell routing mode in
19 which bytes read from and written to the control store have
20 the format depicted in the table;

21 Fig. 31b is a table illustrating the cell function mode
22 showing the byte read from and written to the control store
23 in a particular format;

24 Figs. 32a, 32b, and 32c illustrate rows which are
25 addressed by three different combinations of a row address
26 and a wildcard register value;

27 Fig. 33a, 33b and 33c show the wildcard mask and shift
28 register addressing circuits implemented with a standard
29 address decoder;

30 Fig. 34a is a schematic block diagram of a mask
31 register and alignment unit for use with a wildcard register
32 for use with read cycles;

33 Fig. 34b is a schematic diagram of the internal layout
34 of one of the switches of the mask register shown in Fig.
35 34a;

36 Fig. 34c is a table of state information which depicts
37 bits presented in a right justified form on the external

1 interface following a shift/mask operation;

2 Fig. 35 depicts a table showing the state access bits
3 retrieved in an example access using the separation register
4 technique;

5 Fig. 36a shows an alternative structure to that of Fig.
6 34a;

7 Fig. 36b shows how the circuit of Fig. 34a can be
8 extended to support systems where the width of the input and
9 mask registers is wider than that of the output bus;

10 Fig. 37 depicts a function unit similar to that shown
11 in Fig. 20 which has been modified to support state access;

12 Fig. 38 depicts a structure on which duplicated bits of
13 control store can exist on the same bit line as normal bits
14 of control store;

15 Fig. 39 is a schematic diagram of the relationship of
16 an FPGA, microprocessor and memory device in a typical
17 application;

18 Fig. 40 is a diagrammatic representation of an FPGA
19 architecture showing a 4 bit wide AND gate implemented on an
20 FPGA in accordance with an embodiment of the present
21 invention;

22 Fig. 41 depicts a diagrammatic representation similar
23 to Fig. 40 but of a 16 bit wide AND gate implemented on four
24 4-cell x 4-cell blocks with the 16 cells arranged in one
25 column of the array;

26 Fig. 42 depicts a PAL-type structure showing how the
27 AND plane is structured and mated to an OR plane to form a
28 general purpose logic block;

29 Fig. 43 depicts a one bit accumulator structure from a
30 row of five cells in two 4-cell x 4-cell blocks;

31 Fig. 44 depicts a three bit accumulator with a look
32 ahead carry;

33 Fig. 45 depicts a 16 bit accumulator with a carry look
34 ahead which is a more complex arrangement but similar to
35 that shown in Fig. 44;

36 Fig. 46 is a diagrammatic representation of a 4 bit
37 synchronous counter implemented in a single 4-cell x 4-cell

1 block;

2 Fig. 47 depicts a 16 bit synchronous counter realised
3 on four 4-cell x 4-cell blocks having flyover routing
4 resources; and

5 Fig. 48 depicts a 16:1 multiplexer realised in two
6 4-cell x 4-cell blocks, and which is implemented as a tree
7 of 2:1 multiplexers.

8

9 DETAILED DESCRIPTION OF SOME PREFERRED EMBODIMENTS

10 Fig. 1 depicts an enlarged plan view of part of an FPGA
11 10 in accordance with a preferred embodiment of the
12 invention. A plurality of cells 12 are arranged in a 2-
13 dimensional array in orthogonal rows and columns on the
14 chip. Cells 12 are arranged into 4-cell x 4-cell blocks 14,
15 and the 4-cell x 4-cell blocks are arranged into a 16-cell x
16 16-cell block 16. Cell block 14 is defined by routing
17 switches 18 and 20 at its cell boundaries. Another block 14
18 can be defined by routing switches 18 at all boundaries. It
19 will be seen from Fig. 1 that there are two types of such
20 routing switches: routing switches 18 which form a boundary
21 between 4-cell x 4-cell blocks, and routing switches 20
22 which form the boundary between 16-cell x 16-cell blocks.
23 The routing switches 18 and 20 provide connections between
24 various wires and cells 12. The structure and function of
25 these different types of routing switches will be disclosed
26 in detail later.

27 Fig. 2 depicts an enlarged portion of the area 21 shown
28 in broken outline in Fig. 1 with all the routing resources
29 shown. As will be explained with reference to Figs. 2, 3, 4
30 and 5, there are three main ways by which cells may be
31 connected to each other. As best seen in Fig. 3, at the
32 first level neighbouring cells are interconnected to each
33 other by neighbour interconnects 22 and 24. This is the
34 structure of the above mentioned CAL I array. As shown in
35 Figs. 2 and 4, in each 4 x 4 block 14, additional wires or
36 conductors 26, 28, 30, and 32, hereafter referred to as
37 length-4 flyovers, are routed between the neighbour

1 interconnections. For each row or column having 4 cells
2 there are two length-4 flyovers. In Fig. 4, each pair of
3 flyovers for each row or column of cells is shown on one
4 side of the cells, whereas in Fig. 2 the flyovers are
5 located on either side of a row or column of cells. In the
6 top-most row of cells shown in Figs. 2 and 4 in cell block
7 14, length-4 flyovers 26 and 28 are provided. Length-4
8 flyover 26 conducts signals in the East direction and
9 length-4 flyover 28 conducts signals in the West direction.
10 Vertical flyovers 30 (North) and 32 (South) are provided for
11 each column of 4 cells x 4 cells so that each cell 12 in the
12 4 x 4 array not only has neighbour interconnects 22, 24 but
13 can also interconnect to any of the length-4 flyovers 26,
14 28, 30 and 32. From Fig. 2 it will be seen that
15 horizontally arranged flyovers 26 (E) and 28 (W) are
16 interconnected between switches 18 and 20 as are vertical
17 flyovers 30 (N) and 32 (S).

18 Each 4 x 4 block of cells 14 with the length-4 flyovers
19 shown in Figs. 2 and 4 can itself be considered as being a
20 repeatable cell. One can form an array of these cells each
21 of which communicates with its near neighbours as in Fig. 4.
22 At each position in this array there are two length-4 wires
23 and a length-1 wire, the pair passing to the neighbouring
24 cell in opposite directions. Thus, this array of 4-cell x
25 4-cell blocks has four directed wires for providing output
26 to its nearest neighbours, four wires for receiving signals
27 from its four neighbors, and four wires for receiving input
28 from the length-4 wires directed in the four direction.

29 In the same way, for a 16 x 16 array length-16 wires
30 34, 36, 38, and 40 can be added in the same way as the
31 length-4 wires, as is shown in Figs. 2 and 5. Although
32 Fig. 2 represents only a part of the 16 x 16 array shown in
33 Fig. 1, it includes the length-16 flyovers 34, 36, 38, and
34 40. These length-16 flyovers are both horizontal and
35 vertical. An illustration of this is best seen in Fig. 5 of
36 the drawings, which depicts 4-cell x 4-cell blocks and is at
37 a higher level than the arrangement shown in Fig. 4. Since

1 each block includes four rows of cells and four columns of
2 cells, there are four East length-16 flyovers 34 for each
3 row of blocks, one length-16 flyover for connecting each row
4 of cells in the block. The same is true for the West,
5 North, and South length-16 flyovers. From Fig. 2 it will be
6 seen that the horizontal and vertical length-16 flyovers 34,
7 36, 38, and 40 are inputs and outputs of the larger boundary
8 switches 20 and inputs to the smaller 4-cell x 4-cell
9 boundary switches 18. There is no direct connection from
10 any of the length-16 flyovers to an individual cell 12.

11 It is clear that this process can be repeated with the
12 larger switches 20 of the 16-cell x 16-cell blocks. For the
13 16-cell x 16-cell blocks, in switches 20, three wires are
14 provided in each direction, 3 exiting East, 3 exiting West,
15 3 entering East, and 3 entering West, for example. The next
16 step would be 64 x 64 cell blocks in which switches not
17 shown would have 4 connections in each direction between
18 neighbouring blocks. The arrangement described above
19 defines a hierarchical cellular array built on top of the
20 simple array shown in Fig. 3, with a variety of routing
21 resources whose lengths scale by a factor of 4 each level of
22 the hierarchy. If the level of the hierarchical array is
23 represented as L , with $L = 0$ for the cellular array shown in
24 Fig. 3 (neighbour interconnections only), and the scale
25 factor applied to the array at each stage of the hierarchy
26 is represented as S , which in this example is 4, the flyover
27 wire lengths and the array block sizes (the side length of
28 the blocks) in basic cells for a given hierarchical level
29 are given by S to the power L . For example, $4^0 = 1$ for the
30 neighbour interconnect array. Except for the highest-level
31 switches, there will be $2L+1$ wires in each direction for a
32 total of $2(2L+1)$ wires between blocks on a level L boundary
33 switch. At the highest level, the switch does not have
34 wires connecting to a higher level, so there are $2(2L)$ wires
35 entering or exiting that boundary switch. Normally, the
36 width and height of an FPGA chip in cell units will be an
37 integer multiple of S^L , where L is the highest level of

1 interconnect. Note that for $S=4$, the maximum number of
2 levels = \log_4 chip width, assuming chip width is equal to
3 chip height, but it may be convenient to provide fewer
4 levels. It should also be clear that while $S=4$ appears to
5 be particularly attractive for the scale factor, other
6 values can be used, for example, $S=2$, $S=3$, $S=5$, and $S=8$. It
7 will also be appreciated that the process of hierarchical
8 scaling can be applied to base cells with different
9 connections or can be started with a cluster of cells
10 representing the basic repeating unit at level 0, i.e. the
11 arrangement as shown in Fig. 3.

12 The provision of different levels of routing resources
13 allows the number of segments required to route point to
14 point (cell to cell) on the FPGA array to be minimised. If
15 a straight line path is considered between two points or
16 cells on the array with neighbour connect only, then the
17 number of routing segments would equal the number of cells
18 between the two points. In contrast, as best seen in Fig.
19 6, the hierarchical interconnect results in the number of
20 segments being proportional to the logarithm of the distance
21 between the origin or source and destination cells (plus a
22 few local segments). For example, for a source at cell 2
23 travelling to cell 12 via cell 3, the effective distance via
24 flyovers on the hierarchical routing arrangement is 5,
25 whereas the neighbour cell routing distance is 10. Fig. 7
26 shows the 64-cell x 64-cell structure of a chip which
27 embodies the CAL II architecture. Note that length-64
28 flyovers (level 3) are not provided since with only a single
29 64-cell x 64-cell block they do not provide a significant
30 reduction in routing delays.

31 With the CAL II structure all the wires in the array
32 are directed, and have a single source. Therefore, 3-state
33 drivers are not used. In addition, it will be understood
34 that in the array the connections are completely
35 symmetrical, that is to say that if the array is rotated or
36 reflected, the structure remains unchanged. Both these
37 properties provide considerable advantage. Firstly, multiple-

1 source wires, although allowing additional flexibility, can
2 result in a considerable area overhead, and experience
3 contention where different drivers attempt to place opposite
4 values onto the wire. Contention causes heavy power
5 dissipation and can result in device failure. This is
6 obviated by the single source arrangement of CAL II.
7 Secondly, symmetry of the array simplifies the CAD software
8 which maps user designs onto the array and allows
9 hierarchical blocks of those designs to be rotated or
10 reflected in order to provide better utilisation of the
11 array. It will be understood however that the principle of
12 hierarchical scaling can be successfully applied to arrays
13 which are not symmetrical. It should also be understood
14 that although the previous discussion referred only to
15 single wires, the hierarchical scaling technique is equally
16 applicable where wires are replaced by multi-bit buses
17 running between the blocks.

18 Fig. 8 depicts a block diagram of a 64 x 64 CAL II
19 array. Fig. 8 is essentially a floor plan of the array and
20 demonstrates that there are a row decoder 40 and column
21 decoder 42 for addressing the RAM control store and a total
22 of 128 I/O lines. In addition, there are buffers 44 for
23 global signals and a global I/O 46 associated with the
24 global buffers. There are 15 address and 32 data lines as
25 well as the 128 I/O lines. In addition, there are four
26 global inputs G1 to G4, a reset input and an FHZ input which
27 forces all user outputs to a high impedance state.

28 Fig. 9 is a logic symbol for the CAL II array shown in
29 Fig. 8 and depicts the programming interface for static RAM
30 using chip enable \overline{CE} , chip select, \overline{CS} and read/write \overline{Wr}
31 control signals. The \overline{CE} signal starts the programming mode,
32 and normally it will be fed to all chips in a large array.
33 The \overline{CS} signal may be used to address a single chip in an
34 array of chips such as the 4096 cell embodiment of the
35 CAL II architecture, and read or write data to the addressed
36 chip. Timing on these signals is compatible with common
37 SRAM parts such as the HM 628128 (Hitachi) with a 50 ns

1 cycle time. The SRAM programming interface is supplemented
2 by additional hardware resources designed to minimise the
3 number of processor cycles required for reconfiguration.
4 These resources are initially inactive so that the device
5 looks exactly like an SRAM on power-up.

6 Fig. 10 is an enlarged schematic view of one of the
7 cells 12 of the FPGA shown in Figs. 1 and 2. Firstly, the
8 cell 12 is shown as having 8 neighbour interconnects; 2 to
9 each designated North, South, East and West cell. In
10 addition, cell 12 is also connected to the East and West
11 flyovers 26 and 28 and the North and South flyovers 30 and
12 32 respectively. Within cell 12 is a function unit 48, and
13 also within cell 12 are various switches 50, 52, 54 and 56
14 for receiving signals from the respective neighbour
15 interconnects. The SELF output of function unit 48 can be
16 connected to lines Nout, Sout, Eout, and Wout through
17 multiplexers 50, 52, 54, and 56, respectively. Also, in
18 cell 12, function unit 48 receives input from three
19 multiplexer switches 58, 60 and 62, which receive inputs
20 from neighbouring cells and flyovers and which generate
21 three outputs, X1, X2 and X3, respectively.

22 Function unit 48, one of which is present in each cell
23 12, is capable of implementing any combinational function of
24 two boolean variables, A and B. Additionally, function unit
25 48 can implement one of several three-input functions,
26 namely a two-to-one multiplexer with true or inverted inputs
27 or a D-type edge triggered flip-flop with true or inverted
28 clock, data and reset inputs. These functions are
29 illustrated in Fig. 11.

30 As indicated above, each cell 12 has four neighbour
31 inputs and four inputs from flyovers. From Fig. 10 it will
32 be seen that any neighbour input can be connected to any of
33 the function unit inputs X1, X2 and X3 via multiplexers 58,
34 60, and 62, and then to the cell neighbour outputs Nout,
35 Sout, Eout, and Wout via programmable multiplexers 50, 52,
36 54, and 56. The cell function unit output is available to
37 an external device by reading a bit in the control store.

1 This allows applications to read results from CAL II
2 computations. In addition, register values within circuits
3 implemented on the array can be set by writing to the
4 control store. Vertically adjacent cells have state access
5 bits connected to adjacent bit lines in the RAM: this allows
6 registers of the user's design implemented in the CAL II
7 array to be read and written 8, 16, or more bits at a time
8 according to the selected width of the data bus. Thus, it
9 will be appreciated that the CAL II array architecture
10 effectively supports the "active-memory" model of
11 computation in which operands can be written to memory
12 locations and results read from memory locations.

13 The routing associated with a cell is best described
14 with reference to Figs. 12, 13 and 14 of the accompanying
15 drawings. Fig. 12 shows the basic routing resources
16 provided in cell 12, as shown in Fig. 10. The designation
17 "SELF" refers to the output of function unit 48, which
18 implements the logical operations required by the design.
19 It will therefore be seen that within the cell the routing
20 requires four 4:1 multiplexers 50, 52, 54 and 56
21 respectively for routing the signals into Nout, Sout, Eout,
22 and Wout respectively. Each of those multiplexers receives
23 the SELF signal from function unit 48 and therefore each
24 multiplexer can route signals from one of the other three
25 directions, that is for multiplexer 56 (Wout), output can
26 come from either North, South, East or SELF. Likewise, the
27 other multiplexers can select from the various other inputs
28 to provide the respective output. The implementation of
29 these multiplexers in CMOS technology is disclosed in U.S.
30 Patent 5,243,238 (the CAL I patent); all the other
31 multiplexers in the array can be implemented using this
32 technique.

33 At the next level in the hierarchy, that is, at the
34 junction of the 4-cell x 4-cell blocks, another switching
35 function must be provided. This is best seen in Fig. 13,
36 which depicts a switch 18. Fig. 13 shows the potential
37 inputs and outputs of switch 18. Switch 18 handles only the

1 horizontal East/West going signals, but the switches for the
2 vertical signals are identical. Switch 18 has six inputs;
3 two inputs from cells 12, two inputs from the length-4 East
4 and West flyovers and two inputs from the East and West
5 length-16 flyovers. Because the 4 x 4 boundaries occur at
6 the same position in the array as neighbour boundaries, it
7 is advantageous for switch 18 to include the direct
8 neighbour connections as well as additional connections to
9 longer wires. Including both neighbour and longer wires
10 allows a design which uses only neighbour connections to be
11 mapped onto the hierarchical array without using additional
12 switching units. In general, the outputs of a switch at
13 level 'L' in the array will be a 'superset' of (that is,
14 will include) outputs of a switch at level 'L-1'. As is
15 seen in Fig. 13, it is also convenient to connect the longer
16 length-16 wires into switch 18, which serves lower
17 hierarchical levels. In a larger array using higher levels
18 of hierarchy, length-64 or longer wires may also be provided
19 as inputs to the switch. However, to preserve the hierarchy
20 for switches at 4-cell boundaries, the only outputs are
21 length-1 (neighbour) and length-4 signals.

22 Fig. 14 depicts the inputs and outputs of the switching
23 function on a switch 20 located at the boundaries of the
24 16-cell x 16-cell blocks (see Figs. 1 and 2). Because 16 x
25 16 boundaries also occur at 4 x 4 cell boundaries, then in
26 order to preserve the hierarchy and regularity of the array,
27 the 16-cell boundary switches are arranged to offer all the
28 routing permutations of the 4-cell boundary switches, but
29 also offer additional options related to the length-16
30 wires. This is illustrated by the arrangement shown in Fig.
31 14. In Fig. 14, the hierarchy stops at 16 x 16 blocks and
32 there are no length-64 wires. However, in a larger
33 embodiment, switches using longer wires can be provided.

34 Fig. 15 depicts a preferred switching function for use
35 at 4-cell boundaries. East, West, North and South are the
36 same since the switches are symmetrical, and therefore only
37 the East and West switching functions are shown. Switch 18

1 has two 3:1 multiplexers (that is for the East and West
2 cells) and two 5:1 multiplexers for the East flyover (4) and
3 West flyover (4). With reference to Figs. 1 and 2, it is
4 clear that the switch 18 exists physically at the East/West
5 boundary between blocks. There is a similar switch at the
6 edge of the array to handle connections to the I/O pads, as
7 will be later described with reference to Fig. 26a.

8 Fig. 16 depicts a switching function similar to that
9 shown in Fig. 15 but which accommodates global signals.
10 Switch 18a of Fig. 16 has multiplexers for driving the East
11 and West cells which are the same as those shown in Fig. 12.
12 However, 8:1 multiplexers 72, 74 receive inputs from the
13 neighbouring East and West cells, the length-4 East and West
14 flyovers, the length-16 East and West flyovers, and
15 additionally three inputs from horizontal global signals G1
16 and G2, and a constant 0 signal. Although not shown in the
17 interests of clarity, it will be appreciated that a
18 corresponding switch used for vertical signals will have two
19 vertical global signals G3, G4. Otherwise, switch 18a for
20 the vertical signals will be the same as the horizontal
21 switch 18a. 8:1 multiplexers are preferred so as to allow a
22 straightforward implementation using three RAM-controlled
23 pass transistors. However, in an embodiment in which
24 additional multiplexer inputs are acceptable, four global
25 signals G1, G2, G3, and G4 will all be provided as
26 multiplexer inputs to the flyover multiplexer.

27 It is also desirable that routing delays from one cell
28 to a neighbouring cell across 4-cell and 16-cell boundary
29 switches should be negligible and, consequently, to achieve
30 this the number of sources (inputs) to the multiplexers
31 which connect neighbouring cells must be minimised.

32 Fig. 17 depicts switch 20 located at a 16 x 16 cell
33 boundary. As before, only a switch 20 for an East/West
34 direction is shown, but it will be appreciated that a
35 similar switch 20 is necessary for signal conductors in the
36 North/South direction because of symmetry. Switch 20
37 includes six multiplexers: two 4:1 multiplexers 76 and 78

1 for driving neighbour wires, two 8:1 multiplexers 80, 82 for
2 driving length-4 flyovers, and two 7:1 multiplexers 84 and
3 86 for driving length-16 flyovers. The 8:1 and 7:1
4 multiplexers also receive inputs from the length-4 and
5 length-16 flyovers. In addition to the length-4 switches of
6 the form of Fig. 16, switch 20 includes additional
7 connections for the corresponding multiplexers for driving
8 the length-16 flyovers. In Fig. 17, the BIT signals in
9 multiplexers 84 and 86 are from the RAM bit line.
10 Corresponding North/South switches have a WORD line from the
11 RAM, instead of the BIT line in multiplexers 84 and 86.

12 The switches shown in Figs. 15 through 17 and
13 described above for the 4-cell x 4-cell and 16-cell x
14 16-cell boundaries can be extended for use in larger arrays
15 if required. An appropriate rule for deciding what is
16 connected to each multiplexer in a switch is that a
17 multiplexer outputting a signal at level L in the hierarchy
18 should have inputs from

- 19 1) signals going in the same and opposite directions
20 as the multiplexer output at level L+1 if level
21 L+1 exists,
- 22 2) a signal going in the same direction as the
23 multiplexer output at level L, and
- 24 3) signals going in the same and opposite directions
25 at levels L-1, L-2, etc. down to level 0.

26 It should also be appreciated that this is not the only
27 possible rule and that there are a large number of possible
28 switching functions which can be used in a hierarchical
29 array. Many potential modifications to the switches present
30 here will be immediately apparent to those of ordinary skill
31 in the art. One modification is to include provision for
32 90° turns in the switches by providing inputs from a
33 perpendicular edge (via an additional selector, in
34 conjunction with extra routing resources in the switches
35 themselves, and a different choice of input signals to the
36 multiplexers). Another modification is elimination of the
37 180° wraparound options (for example, eliminating west

1 neighbour input to east length-4 flyover output).

2 The additional routing resources provided by the
3 hierarchical FPGA can be used in four separate ways:

4 1. User Designs at lowest level, FPGA hardware selects
5 longer wires. The user can design using the simple
6 neighbour routing models ignoring the availability of longer
7 connections. Hardware in the FPGA detects when a signal is
8 placed on a series of connections straight through a row or
9 column of cells extending the full length of a flyover, and
10 places the signal on the flyover as well.

11 2. User Designs at lowest level, CAD software selects
12 longer wires. In this case low level CAD software
13 automatically detects nets in the layout which can benefit
14 from the new routing resources and takes advantage of them
15 to speed up the design. With this methodology the flyover
16 wires will often carry the same signal as the neighbour
17 connect at the same position in the cellular array. Adding
18 redundant wiring of longer lengths is easy for the software,
19 and is independent of the number of levels of routing
20 provided by a given FPGA chip, thus providing design
21 portability between different chips.

22 3. User Designs at All Levels, FPGA hardware selects
23 longer wires if usable and control store indicates they are
24 unused. In this case the FPGA hardware uses longer wires
25 when appropriate, but some long wires will have already been
26 taken up by the user. In this case the hardware (an extra
27 NOR gate input, see Fig. 18 discussed below) must detect
28 whether the control store bit has marked the longer wire as
29 unused. In one embodiment, an extra bit is provided for the
30 user to disable the automatic long line selection.

31 4. User Controls Selection of Long Lines. The user
32 can design using a programming model which includes the
33 longer wires, and make explicit assignments to the extra
34 routing resources. In such an embodiment, no automatic
35 selection of long lines by hardware is provided. With this
36 embodiment, CAD software may be selected which optimizes the
37 placement of signals on the various lines in the device.

1 It is advantageous to use the CAD software to detect
2 places in the completed design where wires can
3 advantageously be transferred to longer routing resources.
4 Such occasions can arise where two sub-units of the user's
5 design which themselves must use shorter connections are
6 placed side by side. Providing redundant wiring, for
7 example on both length-4 and length-16 flyovers increases
8 speed. And substituting a longer wire for four shorter
9 wires increases speed and also frees the shorter wires for
10 other uses.

11

12 Hardware Selects Longer Wires

13 The chip may contain special circuitry which can
14 determine when to use longer routing resources. For
15 example, a logic gate can detect when a path through 4
16 neighbour interconnects is used and then automatically route
17 the signals through a length-4 flyover. This hardware
18 option provides more portability. If a company produces new
19 "fast" versions of existing chips by adding longer wires and
20 related logic gates to automatically select the longer
21 wires, existing user designs can be implemented on these
22 faster chips without requiring any changes or effort by the
23 user. In addition, the direct hardware selection of the
24 longer routing wires simplifies the CAD software, allowing
25 it to run faster. Chip hardware which automatically selects
26 faster routes is particularly attractive for applications
27 which dynamically reprogram FPGA chips, where a-priori
28 determination of long routing lines is difficult.

29 Automatic selection of long lines is an extension of a
30 technique disclosed in U.S. Patent 5,243,238, where a NOR
31 gate was used in 4:1 multiplexers to detect the state of two
32 RAM cells and output a 0 corresponding to routing "straight
33 through" (for example, North to South) the multiplexer.

34 Fig. 18 shows a NOR gate circuit which automatically
35 uses a length-4 flyover in response to a user's design which
36 specifies a path through four adjacent neighbour
37 interconnects in a 4 x 4 block. The portion of cell 12-1

1 which detects that RAM cells M1 and M2 both carry logic 0,
2 indicating that a signal from the West is to be routed to
3 the East is illustrated. Corresponding portions are present
4 in cells 12-2 through 12-4, but for simplicity are not
5 illustrated. OR gate OR1 outputs a logic 0 only when RAM
6 cells M1 and M2 both carry logic 0. When corresponding RAM
7 cells for controlling logic cells 12-2 through 12-4 all
8 carry logic 0, four logic 0 signals are input to NOR gate
9 NOR1, causing NOR gate NOR1 to output a logic 1. Switches
10 18-1 and 18-2 show only the circuits which automatically
11 select the length-4 flyover. Multiplexer MX1 located in
12 switch 18-1 is controlled by the logic 1 output of NOR gate
13 NOR1 to place its input signal onto flyover line 134.
14 Multiplexer MX2 in switch 18-2 is controlled to pass the
15 signal on line 134 in response to this logic 1 from NOR gate
16 NOR1. Thus switch 18-2 will provide an output signal OUT
17 faster than if the input signal IN had been routed through
18 cells 12-1 through 12-4.

19 Fig. 19 shows an implementation of NOR gate NOR1 of
20 Fig. 18 available as a single metal wire 191 extending
21 through cells 12-1 through 12-4 into multiplexers MX1 and
22 MX2. The four OR gates OR1 can be physically positioned in
23 their respective cells, so the layout of the NOR gate of
24 Fig. 18 is very compact. The additional NOR gates can be
25 implemented using a standard pull-down technique using an
26 extra wire with transistors which can be pulled down at any
27 one of 4 positions and a p-type device at the end of the
28 wire which acts to pull the wire high in the absence of a
29 pulldown signal. Using several pulldowns and a p-type
30 pullup has the advantage of allowing a distributed layout
31 where the main overhead is a single metal wire routed in
32 parallel with the flyover.

33 The signal from the 4-input gate NOR1 can be fed to the
34 cells 12-1 through 12-4 so that function unit 48 inputs (see
35 Fig. 10) use the flyover inputs rather than the neighbour
36 interconnects (for example multiplexer 58 can be programmed
37 to select its X1 signal from line W4 instead of line W). In

1 a similar way, the technique can be scaled up so that
2 signals travelling on a group of 4 length-4 flyovers under a
3 length-16 flyover running in the same direction can be
4 detected. This is achieved using a further 4-input gate,
5 e.g. NOR gate, which takes as inputs, the outputs of the
6 four 4-input NOR gates NOR1. The output of this further NOR
7 gate can be fed to switches at both ends of the length-16
8 flyover to ensure that the faster length-16 flyover path is
9 used for signal routing. This NOR gate output can also be
10 fed to intermediate length-4 flyovers to allow signals to be
11 taken directly from the faster path.

12 Hardware selection of flyovers is transparent to the
13 user, who obtains the benefit of an automatically faster
14 chip, and results in a simpler programming mode. The
15 technique can be extended straightforwardly to longer wires,
16 i.e. length-64, or length-256 flyovers, for blocks at higher
17 levels of hierarchy. This technique is equally applicable
18 when the flyovers scale by factors other than four. Similar
19 techniques can be used in other areas where a "short-cut"
20 path is provided to supplement routing resources.

21

22 User Control of Wire Length

23 The user can also design using a programming model
24 which includes the longer wires, and make explicit
25 assignments to the extra routing resources. In this case,
26 extra density as well as extra speed is achieved by
27 assigning different nets to the various levels of
28 interconnect at the same point in the cellular array. For
29 example, the CAD software can select a length-16 wire to
30 carry a signal over a sub-unit without disturbing the local
31 interconnect in that sub-unit. With a design style using a
32 programming model which includes the longer wires, and makes
33 explicit assignments to the extra routing resources, blocks
34 of the user design may have to be aligned on 4-cell or
35 16-cell boundaries in the array. Replacing neighbour wires
36 with longer wires can change timing and power consumption,
37 and these changes may be undesirable to the user. To allow

1 user control of long line replacement, an additional bit is
2 added to the control store, and must be set in order to
3 allow automatic addition of longer wires.

4

5 Function Unit 48 of Cell 12, Several Embodiments

6 Fig. 20 is a schematic block diagram of a multiplexer
7 based function unit capable of implementing the function of
8 the function unit disclosed in the corresponding published
9 PCT Application WO90/11648 (equivalent to U.S. Patent
10 5,243,238), but with additional functions. Experience with
11 the CAL 1024 chip which implemented the CAL I architecture
12 indicated it would be desirable to include two new cell
13 functions, a 2:1 multiplexer and a D-register with clear.
14 These are both three-input functions. Use of a multiplexer
15 to speed up carry propagation in cellular logic structures
16 is well known in the literature, for example, Fast Carry-
17 Propagation Iterative Networks, Domenico Ferrari, IEEE
18 Transactions on Computers Vol. C17 No. 2, August 1968 and
19 European patent application serial no. 91 304 129.9, owned
20 by Xilinx, Inc., entitled "Logic Structure and Circuit for
21 Fast Carry", published 13 Nov 1991 under publication no. 0
22 456 475 A2. The multiplexer function is useful for building
23 adder and counter circuits, and a D register serves many
24 user's expressed preferences for TTL like D-registers
25 instead of latches. Both functions were considered for the
26 original CAL 1024 part. One problem with including three
27 input functions in the CAL 1024 architecture was that only
28 the four neighbour inputs could be selected as function
29 inputs. This meant that for a three input function, inputs
30 had to come from three of the four neighbour directions,
31 which is hard to achieve without using adjacent cells for
32 routing alone. Use of a cell for routing alone reduces
33 density. With the new routing architecture of CAL II, it is
34 attractive to allow the length-4 flyovers to be used as
35 function unit inputs, providing a total of eight possible
36 inputs. The additional flyover routing resources mean that
37 the three input functions can be used and density

1 maintained.

2 Fig. 20 shows an embodiment of function unit 48
3 depicted in Fig. 10 which can easily support a 2-input
4 multiplexer and a D-register with clear. Function unit 48
5 consists of the three 8:1 multiplexers 58, 60, 62 each of
6 which receives eight inputs. The eight inputs are from the
7 four immediately neighbouring cells plus the North (N4),
8 South (S4), East (E4) and West (W4) length-4 flyovers and
9 the multiplexers 58, 60 and 62 provide outputs X1, X3 and X2
10 respectively. The three outputs, X1, X2 and X3, are fed to
11 2:1 multiplexers 94, 96 and 98 respectively which provide
12 conditional inversion of X1, X2, and X3. Three further
13 outputs Y1, Y2 and Y3 are created which are fed to a further
14 2:1 multiplexer F which is controlled by the output of the
15 Y1 multiplexer. It will therefore be appreciated that the
16 function unit 48 is based on the 2:1 multiplexer F which is
17 the only multiplexer in the cell which is controlled by a
18 data signal (output of Y1) rather than by the control store.
19 As well as implementing all boolean function of two
20 variables, the 2:1 multiplexer F can be used directly with
21 true or inverted input variables. As will be later
22 described, the 2:1 multiplexer function is useful in a wide
23 variety of circuits including adder carry chains. Fig. 11
24 shows the two-input boolean functions and the three-input
25 (two data, one control) multiplexer and three-input D-
26 register functions implemented by Fig. 20. Generating
27 combinational logic functions using 2:1 multiplexer based
28 function units is well known and is disclosed in the CAL I
29 application. Function unit 48 includes a D-type edge
30 triggered flip-flop 100 which allows it to achieve certain
31 logic functions. D-register 100 receives as a clock input
32 the output Y1 from multiplexer 94, as a clear input the
33 output Y3 from from multiplexer 96, and as data input the
34 output 42 from multiplexer 96. In another embodiment, flip-
35 flop 100 may have an enable input connected to Y3 and no
36 clear input. Alternatively, a clear input may be provided
37 and connected to a special global clear signal fed to every

1 cell. The F multiplexer receives these same three signals,
2 as shown in Fig. 20. The output of the F multiplexer and
3 the output (Q) of D-flip-flop 100 are fed to a further 2:1
4 multiplexer 102, the output of which is designated as the
5 "self" output. The path through multiplexer 98 to the
6 function unit output has been optimised for speed and, where
7 possible, signals on the critical path in the user's logic
8 use this input in preference to X1 or X3.

9 With the function unit shown in Fig. 20 the three-input
10 function capabilities over the existing CAL 1024 chip
11 function unit are achieved. This is in part due to the
12 symmetry of the inputs to the function unit 48 because any
13 of the neighbour and length-4 flyovers can be selected as
14 sources for each of the function unit inputs as can be seen
15 by inspecting multiplexers 88, 90 and 92. Two drawbacks of
16 this function unit 48 are a relatively large delay caused by
17 the 8:1 multiplexers and the fact that a constant 0 cannot
18 be forced onto the clear input of the register to produce a
19 non-clearable flip-flop. Instead a source for clear input
20 must be found which, for example, can be from a global
21 signal via one of the switch units on the 4 x 4 cell block
22 boundaries or by using an adjacent cell to form a constant
23 0. This is a further advantage of having constant 0 as a
24 source for the length-4 flyover wires.

25 Fig. 21 depicts an alternative function unit 114 which
26 can serve as function unit 48 in all of cells 12 shown in
27 Fig. 10 and Fig. 2. Function unit 114 provides a constant
28 source for the register clear input and requires the same
29 number of bits of control store RAM as the Fig. 20
30 embodiment. One more bit controls multiplexer 122 and one
31 less bit controls multiplexer 118. But function unit 114 is
32 less symmetrical in its input selection than the function
33 unit of Fig. 20, as can be seen from an inspection of
34 multiplexers 116, 118 and 120. Lack of symmetry in function
35 unit 114 complicates the software which implements a design,
36 thereby making it more difficult to make effective CAD
37 tools. As before, multiplexers 116, 118 and 120 provide the

1 three outputs, X1, X3 and X2 respectively. The structure is
2 otherwise the same except for the inclusion of a 4:1
3 multiplexer 122 between multiplexer 118 and 2:1 multiplexer
4 F. The output multiplexer 124 receives outputs from the F
5 multiplexer, still controlled by Y1, and the D-type flip-
6 flop 121.

7 A further alternative version of a function unit 126 to
8 be used as function unit 48 in cell 12 is depicted in Fig.
9 22. In this design, the symmetry of the input 4:1
10 multiplexers 128, 130 and 132 is further reduced. The
11 attractions of Fig. 22 are the lower fan-in on the length-4
12 wires which are connected to 1 multiplexer per cell rather
13 than 3 per cell, and the fast X2 path through the function
14 unit itself which improves performance over the function
15 unit of Fig. 20.

16 Fig. 23 shows a further variation on the function unit
17 of Fig. 20 which implements the same operations. The
18 additional 2:1 multiplexers 95 and 97 are controlled by the
19 same bit of RAM which provides for routing the register to
20 the self output via the combinational function multiplexer
21 F, and the 2:1 multiplexer 102 is deleted. The advantage of
22 this unit is that the number of multiplexers between input
23 and output for combinational functions and the X3 input is
24 reduced from 4 to 3 while all other paths still require 4
25 multiplexers, thus X3 provides a fast path through the
26 function until without requiring more control store RAM than
27 the function unit of Fig. 20.

28 The performance of multiplexer based switching
29 structures such as those in Figs. 18, 21, 22 and 23 can be
30 improved using standard techniques at the expense of
31 increasing the area required to implement them. However,
32 because of the area cost it is not considered desirable to
33 increase area of more than a small sub-set of multiplexers.
34 In the network of logic gates in a user's design, a critical
35 path corresponding to the longest signal delay between input
36 and output can usually be identified. To improve the
37 performance of the network of logic gates it is necessary to

1 reduce the delays along the critical path; reductions in
2 delay elsewhere will have no effect. CAD software tools are
3 available which can automatically determine the critical
4 path through a block of combinational logic and implement
5 the critical path using fast path hardware to reduce delay.

6 Fig. 24 depicts an implementation of a gate-based
7 function unit indicated by reference numeral 133. In this
8 case, there are two 8:1 multiplexers 134 and 136, which
9 provide outputs X1 and X2 respectively. X1 and X2 are
10 optionally inverted, then 4:1 multiplexer 138 selects one of
11 the four functions of the resulting variables, that is AND,
12 OR, XOR or DREG. Function unit 133 has only two input
13 variables similar to that of the function unit used in the
14 CAL I FPGA as described in WO90/11648 (U.S. Patent
15 5,243,238) and, consequently, it cannot implement a 2:1
16 multiplexer as a cell function. This design shows that
17 multiplexer based function units are not the only possible
18 way of implementing functions in the cellular array.
19 Another possibility would be to use a 4 bit RAM lookup table
20 addressed by X1 and X2, as discussed in U.S. Patent
21 4,870,302, reissued as U.S. Reissue Patent Re34,363,
22 invented by Ross Freeman, entitled "Configurable Electrical
23 Circuit Having Configurable Logic Elements and Configurable
24 Interconnects".

25 Fig. 25 depicts a further gate-based version of a
26 function unit 140 to implement function unit 48. Function
27 unit 140 is somewhat similar to that shown in Fig. 23 except
28 that 4:1 multiplexer 138 is combined with the neighbour
29 routing multiplexers which become 8:1 multiplexers 142, 144,
30 146 and 148. In addition, 8:1 multiplexer 145 provides an
31 X3 output which passes through an inverter 150. There is
32 also a 2:1 multiplexer 152 to generate a Y3 output which can
33 be fed to the D register, and a 2:1 multiplexer 156 which is
34 controlled by data signal Y3 rather than the control store.
35 Thus, five function outputs Z₁, Z₂, Z₃, Z₄ and Z₅ are
36 generated. Therefore, in this structure cell 12 can compute
37 several different functions of its input variables (X1 to

1 X3) simultaneously and route them to neighbour outputs
2 (Nout, Sout, Eout, Wout). Offering several outputs is
3 advantageous for important functions like adders, but
4 nevertheless requires extra control memory and more chip
5 area, thus function unit 140 is harder to design with.

6

7 One Fast Path

8 The version of function unit 48 shown in Fig. 20 can be
9 constructed in such a way that the path between one of the
10 input variables X1, X2 or X3 and the function unit output
11 SELF is optimised for speed using standard techniques while
12 the other paths are not. Such a function unit 48 requires
13 significantly less area than a function unit 48 in which all
14 input paths are optimised. Software may be written so that
15 signals on the critical path in a user design can be
16 directed where possible to use the optimized X2 input to the
17 function block, ensuring that critical path signals incur
18 minimal delay. This may be done by having the software make
19 selective changes to the user design taking advantage of the
20 symmetrical nature of the function unit which allows inputs
21 of combinational functions to be permuted. Fig. 11 shows
22 logic functions of A and B which are available from the
23 embodiment of function unit 48 shown in Fig. 20. For
24 example, function $X1 \cdot \overline{X2}$ with $X1 = A$ and $X2 = B$, where A is
25 on the critical path can be transformed to $\overline{X1} \cdot X2$ with X1
26 equal to B and X2 equal to A by making changes to the
27 sources of X1, X2 which drive function unit multiplexers 58,
28 60, and 62 in the local cell. Such a technique allows most
29 circuits to obtain similar performance to that available
30 from a function unit where the delays through X1 and X2 are
31 both equally fast, but with much less area overhead.

32

33 Input/Output Structure

34 Fig. 26a depicts a schematic block diagram of the
35 input/output architecture of the embodiment of the CAL II
36 array illustrated in Fig. 8. The circuit of Fig. 26a occurs
37 on the east side of the chip. At the edge of the array of

1 cells 12 there are programmable input/output (I/O) blocks
2 110. Each I/O block 110 is connected to an external pad.
3 Three bits of control store RAM are provided to each I/O
4 block for selecting input threshold voltage (LEVEL),
5 selecting pad slew rate (SLEW), and providing an input pull-
6 up resistor. Flexibility is increased by using additional
7 control store bits to get additional control over pad
8 parameters in this case slew rate and threshold voltage
9 level, or to provide switchable pull ups.

10 There is one I/O block 110 for every two cells 12-A and
11 12-B along the West and East edges of the chip and also one
12 external pad for every two cells along the North and South
13 edges of the chip. This arrangement has the advantage of
14 reducing cost by reducing the number of package pins
15 required. Normally, wide (16-32 bit) buses are connected to
16 the West and East edges as shown in Fig. 8 so that the chip
17 registers latching these buses will be arranged vertically
18 (as will be later described in detail) and hence are
19 efficiently accessible through the control store interface
20 by the host processor. Many variations on this allocation
21 of pads are possible, including providing one I/O block and
22 pad per external cell position on all edges and putting one
23 pad per cell on two adjacent edges and one pad per two or
24 four cells on the other two edges.

25 With regard to the architecture depicted in Fig. 26a,
26 each I/O block 110 has a data input (OUT) and an enable
27 input (EN), each of which are connected directly to cells 12
28 on the periphery of the CAL II array. Similarly, I/O block
29 110 can provide on its IN line a signal to cell 12-A or cell
30 12-B or on West length-4 flyovers W4B or W4A or West
31 length-16 flyovers W16B or W16A. Likewise, I/O block 110
32 can receive on its OUT line a signal from East flyovers E4B,
33 E4A, E16B, or E16A as well as from cell 12-A or 12-B.
34 Thus, the data input to I/O block 110, which is a pad output
35 (labelled OUT in the I/O block) receives data from switch
36 112 and is enabled by the EN output of switch 112. This
37 design minimises delays between the internal array and off-

1 chip by eliminating separate I/O control logic signals. By
2 placing suitable values on the data and enable input signals
3 (which could be achieved by using constant cell functions 0
4 and 1), I/O block 110 can be programmed to operate in input,
5 output, bi-directional, or open drain mode, as shown on the
6 I/O block mode table of Fig. 26b.

7 Fig. 26d shows one embodiment of switch 112 of Fig.
8 26a. Eight multiplexers are provided, as shown, Signal
9 lines and flyover lengths are labeled as shown in Figs. 15-
10 17. Thus, the details of Fig. 26d, which reference the same
11 signals, are not described further. Supporting all 6 input
12 signals from the two cells allows data and enable signals to
13 be sourced from either cell, making it less likely that a
14 pad will not be used because of routing constraints.
15 Additional inputs are provided, in particular, constants 1
16 and 0, and a bit line as inputs to the enable multiplexer.
17 The constant values are particularly useful for the enable
18 signal when the pad is to function as an input (enable=0) or
19 as an output (enable=1) rather than a bidirectional pad.
20 Constant values on the data signal and a computed value on
21 the enable signal produce open drain pull-up (in=1) or pull-
22 down (in=0) pads. However, the I/O architecture of the CAL
23 II array has been designed to minimise input and output
24 delay by eliminating extra pad control logic and,
25 consequently, represents a considerable simplification over
26 the CAL I pad control architecture.

27 The I/O routing switches for the North, South, and West
28 sides of the chip are derived in the same way as the East
29 routing switch shown in Figs. 26a and 26d.

30 In addition to I/O signals, input and output signals at
31 positions which do not have associated external I/O pads can
32 be connected to programming signals on the FPGA (data bus,
33 address bus, mode, read/write, \overline{CE} , \overline{CS} , FHZ). This allows
34 circuits implemented on the FPGA to monitor these important
35 external signals. Thus, a circuit implemented on the FPGA
36 can detect external reads and writes to the status or
37 control memory and automatically clock itself to process an

1 input value or produce the next output value.

2 When global signals are provided, they can be driven
3 from logic signals at the edge of the array in the same
4 manner, as well as from dedicated input pads, as is
5 conventionally the case with the CAL I arrangement. Thus
6 use of logic connections from the edge of the array
7 increases the flexibility of the device and allows
8 additional chip logic to be eliminated, further reducing the
9 part count.

10 Fig. 26c shows an example circuit in which a global
11 signal on line 205 can be taken from four possible sources
12 by a user programmable multiplexer. A programmable
13 multiplexer may be used to select between various potential
14 sources for global signals so the number of potential
15 sources can be larger than the number of global clock lines
16 in the array. First, multiplexer 207 can take an external
17 signal such as a clock signal, which is applied to pad 206,
18 buffered by input buffer 203 and provided to multiplexer
19 207. Second, multiplexer 207 can select an internally
20 generated power-on-reset (POR) signal on line 204, which can
21 be provided as a result of a voltage disturbance or other
22 reason. A reset signal generated automatically by detecting
23 transitions in the power line to allow user logic to be
24 initialised is particularly valuable for chips which use
25 non-volatile control memory such as a flash EPROM or support
26 a low voltage data retention mode for their RAM control
27 store. Third, multiplexer 207 can select a signal from
28 counter/timer 209, which may include an internal or external
29 oscillator. A programmable counter/timer driven by an
30 external crystal or other clock source can provide a
31 flexible clock to user logic. Fourth, multiplexer 207 can
32 select a signal generated by the user's internal logic and
33 selected by I/O block 110-6 from the output of cell 208, an
34 east length-16 flyover, and an east length-4 flyover. Such
35 a global signal could be a clock driven from a cell output.

36

37

1 Register Access: Control Store Manipulation and FPGA
2 Reconfiguration

3 CAL II supports direct access from the processor to
4 nodes within the user's circuit. The output of any cell's
5 function unit 48 (see Fig. 10) can be read and the state of
6 a cell which is configured to implement a register can be
7 written by the processor. These accesses are done through
8 the control store interface and require no additional wiring
9 lines to be added to the user's design. The row and column
10 signals which address these registers can be selected as
11 sources within a length-4 switch unit, so that user circuits
12 can detect that an access has been made and take the
13 appropriate action, for example, calculate a new value for
14 an output register or process a value to be placed into an
15 input register. In many applications, this access to
16 internal nodes will be the main path through which data are
17 transferred to and from the processor. In some coprocessor
18 applications, it may be the only method used by the
19 processor to access nodes in the FPGA. User programmable
20 I/O pads may not be used at all.

21 To allow high bandwidth transfers between the processor
22 and internal nodes of the FPGA, it is necessary to transfer
23 a complete processor data word of up to 32 bits in one
24 memory cycle. For this reason, register or gate access bits
25 in CAL II are mapped into a separate region of the device
26 address space from configuration bits. Fig. 27a shows the
27 mapping of this area of the address space. In an embodiment
28 of the CAL II architecture having 4096 cells, there are 64
29 columns and 64 rows of cells. Since there is only one
30 function unit per cell, one memory cell in the control store
31 is sufficient to represent one cell function unit output.
32 Fig. 27a represents a memory space which the processor can
33 address to access the cell function unit outputs. One cell,
34 cell 12-6-23 (where the 6 designates the row and 23
35 designates the column) is shown as a blown-up cell,
36 equivalent to cell 12 shown in the earlier figures. A 6-bit
37 column address CA[0:5] selects a particular column of cells

1 to access. All bits to be accessed in one memory cycle must
2 be in the same column of cells. Select unit 275 selects a
3 subset of these rows to connect to an external data bus.
4 Several possible implementations of select unit 275 are
5 discussed below in connection with Figs. 28, 34a, 34e, 35,
6 36a, 36b, and the run length register discussion. The
7 advantage of select unit 275 is that the fixed relationship
8 between rows or words in a memory array and lines on a data
9 bus in prior art structures is replaced with a programmable
10 relationship. A row select decoder selects one or more of
11 rows 0 through 63 and reads the programmably selected
12 values, applying them to data bus D[0:8,16 or 32], or writes
13 the data bus value into the selected memory locations.

14 The functional operation of storage manipulation and
15 reconfiguration of data in the array will now be described.

16 Fig. 27b is a diagrammatic representation of a RAM
17 programmed FPGA, which consists of the array of RAM cells
18 160 embedded in which are active logic structures controlled
19 by the neighbouring RAM cells. The logic structures
20 implement signal switching and function generators as
21 required by the application mapped onto the FPGA. Details
22 of the functioning of the RAM cells and their use in the
23 FPGA control store are disclosed in WO90/11648 to Algotronix
24 Limited and Principles of CMOS VLSI Design, A System
25 Perspective, Weste, N. and Eshraghian K., published by
26 Addison Wesley 1985.

27 In the structure shown in Fig. 27b it will be seen that
28 there is a data bus 162 and an address bus 164. Each row
29 166a, 166b etc. of RAM cells 160 is connected to the data
30 bus 162 via a row decoder 168a, 168b, etc. Address bus 164
31 is similarly connected to each column 170a, 170b, 170c
32 etc. of RAM cells via a column decoder 172a, 172b, 172c etc.
33 The lines interconnecting columns of RAM cells 160 are
34 termed word lines and similarly the lines connecting
35 horizontal rows of RAM cells with row decoder are termed bit
36 lines. When an address is applied to the RAM array shown in
37 Fig. 27b, a single word line and a single bit line for each

1 bit of the data bus will be activated. Since bits of a word
2 are in a vertical line, addressing a RAM cell results in a
3 vertical vector (column) of RAM cells being written.

4 Reconfiguration time for dynamically programmed FPGAs
5 is an overhead which reduces the potential advantage in
6 computation time over conventional computers. It is
7 therefore essential to minimise reconfiguration time if
8 dynamic reconfiguration is to become practical for user
9 applications. It is also essential to minimise
10 reconfiguration time to reduce the cost of device testing
11 where a large number of configurations is required.

12 If a single word line is active then a narrow data bus
13 is a limiting factor on the number of write cycles required
14 to configure the array. Consequently, narrow data bus width
15 limits or restricts the configuration time. Making the data
16 bus width identical to the number of row decoders enables an
17 entire column of RAM cells to be written simultaneously. In
18 this case, the row address is redundant. In the case of the
19 CAL I array this would require a data bus 128 bits wide and
20 hence require 128 external pads for maximum throughput. It
21 will be appreciated that FPGAs have a large number of logic
22 I/O pins from the array (in the case of the CAL 1024, 128
23 pins) so if the data bus pins are shared with logic I/Os,
24 wide data buses can be supported. Although one data bus bit
25 per row decoder is unfeasible, a system which supports a
26 data bus bit for every two or four row decoders is feasible.
27 Using the I/O pins on the same edge of the chip as the row
28 decoders means that no long wires between pads and row
29 decoders are required. Driving two to four decoders with
30 one I/O pin is especially useful for device testing to
31 minimise the number of vectors required. However, very wide
32 data buses are less useful in actual applications because of
33 the mismatch to the data bus widths of conventional
34 processors and the overhead of board wiring. Using the same
35 pad for both I/O and programming data is also a considerable
36 inconvenience to the designer of boards containing FPGAs.
37 Systems which take advantage of bit line parallel writes by

1 providing block transfer modes such as those becoming common
2 on commercial DRAM chips ("A New Era of Fast Dynamic RAMs",
3 Fred Jones IEEE Spectrum, October 1992) allow high bandwidth
4 for relatively low pinout, and may be attractive for use in
5 future FPGAs.

6 FPGA configurations are usually highly regular compared
7 with the data found in normal data memories. This
8 regularity is especially apparent in the configuration data
9 for user designs implementing computational applications
10 where the circuits themselves usually consist of vectors of
11 repeating bit slice units. Regularity is also apparent in
12 configurations for testing the device. As depicted in Fig.
13 27c, which shows in more detail the RAM addressing circuits,
14 if the columns of the FPGA device are considered as
15 consisting of a sequence of words the same width as the data
16 bus, each word having a unique row address, then it is
17 likely that many of the values in these words are the same.
18 In the CAL 1024 FPGA (CAL I array) device there are 16 such
19 words and in a typical configuration there are an average of
20 3.4 distinct values in the 16 words. This implies that an
21 architecture in which all the words with the same
22 configuration could be written simultaneously could reduce
23 the number of writes required on average from 16 per column
24 to 3.4 per column. Similarly, in a row of 144 words, there
25 may only be 35 distinct values. Thus, an FPGA architecture
26 which activates several word lines simultaneously can also
27 reduce the number of write cycles required. However,
28 activating several word lines simultaneously during a write
29 cycle is more complex because there is a fan-out problem;
30 the buffer at the row decoder must potentially overcome the
31 opposite value stored in several RAM cells on that bit line.
32 This limits the number of word lines which can be active
33 simultaneously, with the exact number depending on a variety
34 of factors, but the number of active word lines is much less
35 than the total number of word lines and a value of 4 is
36 reasonable.

1 Operation of Match Registers

2 One method of advantageously providing or facilitating
3 multiple writes is to replace either the row or column or
4 both address decoders with match registers which compare the
5 value of the address applied to the value stored in the
6 local register. There will be one match register
7 (programmable decoder) where each decoder would otherwise
8 be. If the match register detects that the address matches
9 its stored value, the register activates the corresponding
10 bit or word line, as shown in Fig. 28. Match registers can
11 be programmed to respond to different addresses. Thus,
12 different patterns of registers can be written to
13 simultaneously by programming the same address into a
14 selected group of match registers. In Fig. 28, both row and
15 column decoders have been replaced by match address
16 registers 180a, 180b, etc. for the rows and 182a, 182b etc.
17 for the columns. If the value stored in each register 180,
18 182 is the index of the corresponding bit or word line, then
19 this structure will function as a normal RAM. Functioning
20 as a normal RAM is a desirable state to initialise the match
21 registers to. By storing the same value in multiple
22 registers, the system can be set up to write multiple words
23 when a given address is presented.

24 An additional level of flexibility is provided if the
25 row address decoder is replaced by the structure shown in
26 Fig. 28 where the match register is supplemented by an
27 additional register 184a, 184b which holds a number of the
28 bit of the data bus 162 (which bit in the row) to be used to
29 drive the bit lines when the match occurs. In conventional
30 memories, there is one row address decoder per word and each
31 data bit is connected to a fixed data bus line. However, In
32 Fig. 28, there is one address decoder per bit line, and the
33 mapping to data bus lines is programmable. Thus, there are
34 no fixed word boundaries. This has the considerable
35 advantage of allowing multiple sub-fields to be written
36 simultaneously. The structure of Fig. 28 is considerably
37 more efficient in dynamic reprogramming applications where

1 it is desired to make a change to multiple bit slices.

2 In an embodiment of the CAL II architecture having 4096
3 cells, the format of the address bus is shown in the table
4 of Fig. 29. The first six bits define the cell column, the
5 next six bits the cell row, bit 12 the side, and bits 13 and
6 14 the mode. Smaller CAL II devices have proportionally
7 fewer bits allocated to row and column addresses. The mode
8 bits determine which area of the control store is to be
9 accessed according to the table shown in Fig. 30.

10 When the address bus is in cell routing mode, bytes
11 read from and written to the control store have the format
12 shown in the table of Fig. 31a, which shows the control
13 store byte for programming routing multiplexers. When side
14 = 0 the external routing multiplexers are accessed, i.e.
15 South, West, East or North. In this embodiment, no data are
16 provided for side=1 and mode=cell routing.

17 This control store layout corresponds to function unit
18 126 of Fig. 22. When the address bus is in the cell
19 function mode, bytes read from and written to the control
20 store have the format shown in the table of Fig. 31b. When
21 the address bus is in the channels, I/O mode, bytes read
22 from and written to the control store control the
23 multiplexers in the switches (see switches in Figs. 15-17,
24 and 26d). When the address bus is in the state access or
25 device configuration mode, the state of function units of
26 the device will be read or written (written if the function
27 unit is a flip flop). Control store registers which control
28 shift and mask units, wildcard units, and the state access
29 mask register are mapped into the state access region of the
30 device address space when the "side" bit is set to 0. State
31 access transfers (reading and writing to function units)
32 take place when the "side" bit is set to 1. One additional
33 device control register includes two bits which allow
34 selection of external data bus width, in one embodiment
35 between 8, 16, and 32 bit width. A third bit can force all
36 user I/O pins to high impedance irrespective of the
37 configuration of the I/O control store value for individual

1 pins. This third bit overrides the FHZ signal which forces
2 high impedance on an individual pin. (see Fig. 26d) Upon
3 reset of the device, the data bus width goes to 8 bits and
4 this third bit selects high impedance. During operation,
5 after a valid configuration has been loaded, an external
6 microprocessor will set this bit (select not high impedance)
7 to allow user I/O signals to drive the pins. These tables
8 are included only by way of example to make concepts more
9 concrete. Many other encodings are possible.

10 Although the match register approach allows for maximum
11 reduction in the number of writes to the array, it entails a
12 considerable overhead for setting up the values in the match
13 registers and bit selection registers. For example, if a
14 system with two data lines D0 and D1 is considered, a column
15 of the RAM could be set up with a single write cycle by
16 setting all the match registers on bit lines whose RAMs were
17 to be 0's to select D0 and all the match registers on bit
18 lines whose RAMs were to be 1's to select D1, then writing
19 binary 10. One write cycle per bit line is required to set
20 up the select registers, so this technique is less efficient
21 than the standard RAM interface for configuring the entire
22 array. However, in some computational applications where it
23 is necessary to make one of a smaller number of changes to
24 the control store very quickly (for example, to select a
25 source operand for a computational unit by reprogramming a
26 vector of switches through a control store interface), the
27 match register approach represents an improvement over prior
28 art programming.

29 It is desirable to support multiple simultaneous writes
30 to take advantage of the regularity of the control store
31 (configuration memory) programming data but to minimise the
32 overhead operations required. This can be done by:

33 1. The use of run length registers. A run length
34 register tells how many sequential words are to be written
35 in response to a single address. In this technique the row
36 and the column address decoders are supplemented with
37 additional run length registers. When an address matches

1 the corresponding decoder, the next N decoders, where N is
2 the value stored in the run length register, are enabled and
3 write the word on the data bus onto the bit lines. If the
4 value of 0 is stored in the length registers then only the
5 address decoder is enabled and the device functions as a
6 normal random access memory.

7 The principal advantage of this approach is its ability
8 to configure small rectangular areas of the chip. But a
9 disadvantage is that no implementation has been found which
10 is both small enough to require roughly the same area as the
11 standard decoder and to allow write cycles of the same
12 duration as standard RAMs. However, the technique can be
13 readily implemented by a sequence of single word writes to
14 numerically successive locations. Although this is not as
15 fast as truly parallel writes, it is significantly faster
16 than a sequence of writes under the control of an external
17 processor and would free that processor to undertake another
18 task simultaneously with the writes.

19 2. Wildcard addressing. In this technique the row
20 and address decoders are supplemented with additional
21 wildcard registers which can be written through the RAM
22 interface. The wildcard register has one bit for each bit
23 in the row address. A logic 1 bit in the wildcard register
24 indicates that the corresponding bit in the address is to be
25 taken as "Don't-Care": that is, the address decoder will
26 match addresses independent of this bit. When power is
27 applied, the wildcard registers are initialised to logic 0
28 so that all address bits are treated as significant. Also,
29 the wildcard register is disabled during read operations and
30 when the address bus is in State Access mode. The wildcard
31 register allows many cell configuration memories in the same
32 column of cells to be written simultaneously with the same
33 data. This is used during device testing to allow regular
34 patterns to be loaded efficiently to the control memory but
35 is more generally useful especially with regular bit sliced
36 designs because it allows many cells to be changed
37 simultaneously. For example, a 16-bit 2:1 multiplexer can

1 be built using cell routing multiplexers and switched
2 between two sources using a single control store access.
3 For example, using East routing multiplexers, the two
4 sources could be a cell's function unit output, perhaps a
5 register output, and the cell's West input. When a bit of
6 the wildcard register is active the corresponding bit of the
7 address is ignored, for example if the wildcard register for
8 the lowest order bit holds 01 and the address supplied is
9 10, then decoders 10 and 11 are both enabled. If the
10 wildcard register for the lowest order bit holds 00, then
11 the device functions as a normal RAM.

12 Figs. 32a, 32b, and 32c show three examples of wildcard
13 addressing. In Fig. 32a, a user has not set any bits in the
14 wildcard and has applied the row address 010101 (decimal
15 21). Thus only row 21 is addressed, as shown. In Fig. 32b,
16 the user has set the wildcard address 100001 and again
17 applied the row address 010101. This time, the value 1 in
18 the least and most significant bits causes rows with
19 addresses

20 010100 (decimal 20)
21 010101 (decimal 21)
22 110100 (decimal 52) and
23 110101 (decimal 53)

24 to be addressed. In Fig. 32c, the user has set the wildcard
25 address 000111 and applied the same row address 010101.
26 This combination causes rows 16 through 23 to be addressed.
27 Thus many combinations and densities of multiple rows can be
28 addressed by selecting an appropriate entry in the wildcard
29 register and an appropriate row address.

30 The principal advantage of the wildcard addressing
31 approach is that it can be implemented with standard address
32 decoders without space and time penalties. This arrangement
33 is depicted in detail in Figs. 33a, 33b and 33c. As
34 indicated in the aforementioned application W090/11648 (U.S.
35 Patent 5,243,238) and in the aforementioned Weste and
36 Eshraghian book, the standard address decoder for RAMs
37 consists of a CMOS NOR gate. Both true and complemented

1 values of each address bit, that is A and \bar{A} (eight in this
2 case $A_0\bar{A}_0$ to $A_7\bar{A}_7$) are fed through all gates. Each
3 individual gate selects either the true or the complemented
4 value of the address bit according to the address it decodes
5 by placing a contact on the appropriate metal line. For
6 example, as shown in Fig. 33a, address decoder 183 for
7 decoding 0 uses all true forms (so that if any of the
8 address inputs is high, the corresponding decoder output is
9 low). This is repeated for each of N row decoders. As
10 shown in Fig. 33b, by inserting AND gates controlled by the
11 wildcard register on both the true and complemented signals
12 being fed to the array, both the true and complemented
13 signals for a given address bit can be forced to the low
14 condition. This means that any NOR gate for which the other
15 address bits match will have its output active (high)
16 independent of this bit. Fig. 33b shows part of such an AND
17 gate in the wildcard unit circuit. It will be appreciated
18 that this circuitry is duplicated for each address bit. The
19 E_n signal applied to the AND gate input of Fig. 33b comes
20 from the corresponding bit of the wildcard register.
21 Normally, these AND gates would not be present and the \bar{A}
22 signal would be derived from A using an inverter. Fig. 33c
23 shows that the wildcard unit of Fig. 33b is located within
24 the RAM between the external address bus and the bus to the
25 row and column decoders.

26 As well as being easily implemented, the wildcard
27 address register has an additional important benefit. In
28 many bit-sliced structures found in computational
29 applications, it is desirable to change the same cell in
30 each bit slice unit simultaneously. In addition, it is
31 often the case in fine-grained FPGAs that the cells to be
32 changed simultaneously would be every second, fourth, or
33 eighth cell along a row or column. The wildcard address
34 decoder allows this sort of operation to be performed
35 efficiently by masking out the second, third, or fourth bit,
36 respectively, of the address bus.

1 Testing

2 A further advantage of the wildcard address register is
3 the reduced time required to functionally test the FPGA.
4 Following manufacture it is desirable to test the device to
5 confirm that processing errors or defects have not
6 introduced cells or areas of the FPGA which do not function
7 correctly. Reprogrammable FPGAs are well suited to such
8 testing. The cost of testing is a significant portion of
9 the total manufacturing costs and the cost of testing is
10 almost directly proportional to the number of test vectors.
11 In its most basic form such testing may involve writing then
12 reading particular bit patterns to and from the control
13 store. By comparing the value written with that read back,
14 it can be confirmed that the control store memory is
15 functioning correctly. It is well known that a careful
16 choice of bit patterns can be used to verify correct
17 functioning of the control store with only a small number of
18 test vectors.

19 An alternative and more exhaustive test of the FPGA
20 behaviour would involve the stimulation of every multiplexer
21 with each possible combination of inputs. The procedure for
22 testing the multiplexer behaviour requires that a large
23 number of regular configurations need to be written in order
24 to test each multiplexer. Such a test would involve
25 exercising function multiplexers and also the routing
26 multiplexers.

27 Both control store testing and multiplexer testing
28 involve writing repetitious and regular bit patterns to the
29 configuration memory. Each benefits from the wildcard
30 address register. By using the wildcard address register it
31 is possible to apply a common test configuration pattern to
32 a large number of cells using fewer write cycles than is
33 required when using a conventional memory interface.
34 Similarly, when exercising the multiplexers, the ability to
35 read back function unit outputs from a group of cells
36 provides a substantial reduction in the number of read
37 cycles required. The FPGA testing using wildcard registers

1 thus takes significantly less time to test exhaustively, or
2 alternatively the FPGA could be subjected to a more
3 extensive test in a given time period.

4

5 Shift and Mask Registers

6 It is also desirable to provide access to sub-fields of
7 configuration words. However, this is not achievable using
8 normal RAM addressing or even using wildcard or run length
9 registers. Access to sub-fields is a common requirement
10 because a single word of configuration memory usually
11 contains configuration data for several separate switching
12 units which it is often desirable to change independently.
13 With a word-wide interface, a complex sequence of shift and
14 mask operations is necessary on the host processor in order
15 to change one logical unit without affecting the states of
16 the others. These shift and mask operations often make it
17 impossible to take advantage of the ability to perform the
18 same write operation on several words simultaneously since
19 the bits of the words not to be changed might well be
20 different in each word. This problem is solved by providing
21 a separate mask register and placing a shift and mask unit
22 between the external data bus and the data bus to the bit
23 line drivers as shown in Fig. 33c.

24 A detailed arrangement of a shift and mask register for
25 read cycles is shown in Fig. 34a. For write cycles the same
26 unit as shown in Fig. 34a would be used facing in the
27 opposite direction (that is, its input would come from off
28 the chip and its output would go to the bit line driver data
29 bus) and additional enable lines for each data bus bit would
30 be supplied to the bit line drivers sourced from the mask
31 register. From Fig. 34a it will be seen that the shift and
32 mask register generally indicated by reference numeral 200
33 includes switches 201 placed between the external data bus
34 and the internal data connections 162 to the RAMs 160 (Fig.
35 27b). After power up, register 200 contains all logic 0s
36 corresponding to a conventional RAM interface. Data are
37 loaded into the shift and mask register as part of the

1 control store configuration, or periodically by the
2 microprocessor during reconfiguration. A logic 1 in a
3 particular bit of the mask register indicates that the
4 corresponding bit of the internal data bus 162 is not
5 relevant. On the read operation, only "valid" bits, that is
6 those with a logic 0 in the shift and mask register, are
7 presented in right justified form on the external interface,
8 i.e. at data out. This is depicted by the table in Fig. 35
9 of the drawings. Figs. 34a, 34b and 35 show that each
10 switch 201 has an input from the mask register 200. Switch
11 201 operates as follows: on switch row 7, switch 201-77
12 receives bit b7 as data input InB (see Fig. 34b). Data
13 input InA of switch 201-77 is connected to ground. When
14 mask register bit M7 is high, transistor 203a is on, and the
15 InA input (ground) appears at the output, i.e. bit b7 on InB
16 is masked. The output of switch 201-77, which is ground, is
17 fed to switches 201-66 and 201-67 in row 6. Also fed to
18 switch 201-67 is a ground signal, and fed to switch 201-66
19 is bit b6. Fig. 34c shows that in row 6 there is no enable
20 signal, therefore transistor 203a in switches 201-66 and
21 201-67 stays off and transistor 203b is on, so that inputs
22 b7 and b6 on InB pass to the output. At row 5, again the
23 shift and mask register is not enabled, so signals b7, b6,
24 and b5 pass straight down. At row 4, the mask register bit
25 M4 is enabled, so bit b4 is not shifted down and bits b7,
26 b6, and b5 are shifted to the right and down. This is
27 repeated for other switches depending on the value of the
28 bit in the mask register. Fig. 34c shows the effect the
29 bits in the mask register have on bits passing from Data In
30 to Data Out. It will be appreciated that the shift and mask
31 register simplifies changing isolated multiplexers in the
32 control store. For example, changing a source for a cell's
33 North multiplexer without the shift and mask register would
34 entail the following operations:

- 35 1. Read control store at appropriate address.
- 36 2. Mask out bits corresponding to North register with
37 binary 00111111.

- 1 3. Get new value for North register and align with
- 2 bits 6 and 7. Make sure other bits are 0.
- 3 4. OR new value with value from operation 2.
- 4 5. Write back.

5 Using the shift and mask unit the following steps suffice:

- 6 1. Write mask register with binary 00111111.
- 7 2. Write new value to control store at appropriate
- 8 address.

9 On write cycles the unit disables those bits of the
10 data bus to the bit line drivers corresponding to the 1 bits
11 in the mask register so that no writes are performed on RAM
12 cells in those bit lines. The values in the other bit lines
13 are sourced from input data bus bits in order starting with
14 the least significant bit. This has the advantage of
15 allowing single multiplexers to be written using right
16 aligned data so that the processor does not have to perform
17 additional shift operations. The wildcard register with the
18 shift and mask facility can also be used to allow multiple
19 writes to the same sub-unit of several control store words.
20 For write cycles, the same unit as shown in Fig. 34a is
21 used, but the input comes from off the chip and the output
22 goes to the bit line driver data bus. Enable lines for each
23 external bit are supplied to the bit line drivers sourced
24 from the mask register. It should be apparent that the
25 shift and mask functions are independent, and that shift
26 only and mask only units could easily be derived.

27

28 State Access

29 Current FPGA designs allow read access to the outputs
30 of individual gates and flip-flops of user designs by
31 mapping them into bits in the device control store. Fig. 37
32 shows the additional logic in function unit 48 to support
33 read and write state access. Read and write operations to
34 flip flop 207 are to separate addresses: here write uses Bit
35 0, Word 0 and read uses Bit 1, Word 0. Transmission gate
36 205 is controlled by word 0. For reading, when word 0 is
37 addressed, transmission gate 205 places the output of 2:1

1 multiplexer 301 onto the bit1 line. If the bit1 line is
2 addressed, this value is read. If 2:1 multiplexer 301 has
3 been programmed by its control store bit to pass the Q
4 output of D flip flop 207, this value will be read. For
5 writing to register 207, bit 0 and word 0 are addressed.
6 Register 207 has asynchronous Set and Reset (R,S). AND
7 gates 302 and 303 are connected to Set and Reset
8 respectively. If word0 is 0, AND gates 302 and 303 both
9 carry logic 0, and the value in D flip flop 207 is not
10 changed. If the signal at word0 is 1, a logic 0 on bit0
11 produces a high reset output from AND gate 303, causing D
12 flip flop 207 to store logic 1. Likewise if bit0 is logic
13 0, D flip flop 207 stores logic 0. Similar logic to that
14 provided by AND gates 301 and 303 and transmission gate 205
15 can be applied to other function unit designs such as shown
16 in Figs 18-23. AND gates 302 and 303 need to have non-
17 standard threshold voltages on their inputs connected to bit
18 line bit0. This ensures that in this case that the bit0
19 voltage is at an intermediate value neither logic 1 nor
20 logic 0, the register state will remain unchanged. Such a
21 situation occurs for registers whose word line is selected
22 by the column address but whose bit line is not selected,
23 i.e. the access is to another register on the same word
24 line. Alternatively, complementary bit lines bit0 and $\overline{\text{bit0}}$
25 as used in 6-transistor SRAM cells can be used to avoid the
26 need for nonstandard gates.

27

28 Register Access: User Registers Separated from Configuration
29 Memory

30 Access to gate output and register state is in itself
31 of limited use as an I/O mechanism for communicating data
32 between FPGAs and a host processor. (Access to gate output
33 and register state is, however, very useful for debugging
34 systems.) I/O use has been limited because a large number of
35 overhead shift and mask operations are required to assemble
36 a single word of data from bits taken from multiple RAM
37 cells in the FPGA, possibly from different words in the

1 processor address space. To make gate output and register
2 state access a useful communications interface, it is
3 necessary to provide hardware which allows word-wide read
4 and write accesses from the processor to registers which are
5 part of the user design.

6 The techniques described here to provide access to
7 internal state are perhaps most conveniently applied to an
8 FPGA with a RAM control store because the circuitry can
9 easily be shared with that to access the control store.
10 However, providing internal state access can be used for
11 other types of FPGAs including, but not limited to, anti-
12 fuse and EPROM based structures.

13 In particular, if it is assumed that every computation
14 unit in the FPGA is assigned a single bit within the device
15 control memory to allow read access to gate output and
16 register state and write access to register state, then the
17 first step in improving the bandwidth of the interface is to
18 map bits of RAM which represent register state or gate
19 output into a logically distinct segment of the address
20 space, rather than intermingling them with other
21 configuration memory bits. In one embodiment, register
22 state bits are still physically intermingled with the
23 configuration bits of the array. Segregating the register
24 state bits from the configuration bits is achieved in the
25 simplest way by providing an additional "mode" bit within
26 the address bus as discussed in connection with Fig. 29, and
27 designing the decoders such that the decoders which
28 correspond to the device configuration bits use the true
29 form of the mode signal and decoders which correspond to the
30 state access bits use the complemented form. This
31 segregation results in making the address space less dense
32 but makes it much easier to dynamically change configuration
33 bits or to access state bits. An address bit format is
34 depicted in Figs. 29 and 30. This segregation scheme can be
35 used where the state access bits respond to the same bit and
36 word lines as the configuration bits at the expense of
37 additional complexity in the row and column decoders;

1 thereby each decoder now has two NOR gates corresponding to
2 addresses within the two address spaces and the mode bit
3 selects which NOR gate output is used to enable the bit or
4 word line circuits. It is also convenient to connect the
5 bit line to a different data bus line when the bit line is
6 active in state access mode than when active in
7 configuration access mode. Selecting the data bus is done
8 using extra circuitry in the bit line driver.

9

10 Word-Wide Interface Having Row and Column Separation 11 Registers

12 Given that the state bits are mapped into a logically
13 distinct section of the address space, the best interface to
14 allow word-wide access to internal registers must be
15 considered. Word-wide access techniques could also be
16 applied to access small RAMs in the FPGA such as those in
17 the Xilinx XC4000 system. One reasonable constraint is that
18 bits of registers should occur in order from least
19 significant bit to most significant bit evenly spaced along
20 a single row or column of cells. This constraint is met by
21 most user designs on existing fine-grained FPGAs. With this
22 constraint one can specify an interface using two additional
23 registers which contain row and column separation
24 information. Writing into one of these additional registers
25 would automatically clear the other additional register and
26 determine whether a register was to be accessed along a row
27 or column of the array. One register value would specify
28 the number of cells between register bits. For example, if
29 the data bus width was 8 bits and one accessed address 8
30 with the separation register holding the value 2, one would
31 get addresses 8, 10, 12, 14, 16, etc. This example of the
32 separation register controlling the state access of the bits
33 is illustrated in Fig. 35.

34

35 Use of Wildcard and Shift/Mask Registers to Access State 36 Information

37 While the above interface is, in many ways, ideal, it

1 involves additional logic in registers which cannot easily
2 be shared with the logic required for programming the
3 configuration memory. An attractive option is to use the
4 existing wildcard and bit shift and mask units for accesses
5 to state information as well. While they are not as
6 flexible as the interface using separation registers and
7 require some overhead operations in the processor when
8 registers do not align well with cell addresses in the
9 array, they do provide a significant increase in flexibility
10 over standard RAM access modes. In this context, it may be
11 convenient for the internal data bus to the bit line drivers
12 to be much wider than the external data bus.

13 A variation on the above approach is to use a larger
14 shift and mask register with one bit per cell row. In this
15 case, the row wildcard units are unnecessary for accessing
16 state information. Since the shift and mask register is
17 likely to be significantly wider than the external data bus
18 and the data bus to the bit line drivers, more than one
19 external write operation will be required to set the
20 contents of the shift and mask register. Fig. 34c shows how
21 the circuit of Fig. 34a can be extended to support systems
22 where the width of the input and mask registers is wider
23 than that of the output bus.

24 It is also possible to provide several mask registers,
25 each of which holds a pattern corresponding to a different
26 register of the user's design, and each of which can be
27 conditionally connected to the shift and mask logic. During
28 a register access operation, bits on the address bus can be
29 used to select which of these mask registers is used.
30 Having several mask registers considerably increases the
31 flexibility with which registers can be placed in a user's
32 design.

33 One disadvantage of the shift and mask circuit of Fig.
34 34a is that a signal must pass through a significant number
35 of switches 201 on the path between input and output. This
36 reduces speed of register access operations. Fig. 36a
37 illustrates an alternative shift and mask unit which has

1 only a single switch on each path between input and output,
2 with additional decoding to enable the particular switch.
3 The decoding circuit incurs delay, but this delay is during
4 the write to the mask register, not during access to the
5 user's register. The particular embodiment of Fig. 36a
6 includes 64 mask bits for accessing 64 bit lines, of which
7 no more than 32 will be accessed at one time.

8 As shown in Fig. 36a, a mask register M holds 64 mask
9 bits M0 through M63. A column of 63 incrementers H1 through
10 H63, of which only a few are illustrated, keeps count of the
11 number of logic ones stored in mask register M. Each logic
12 1 causes a corresponding data bit to be provided as data.
13 Circuitry for mask bit M0 is simplest. When M0 is logic 1,
14 transistor T0 is turned on, thus connecting bit line B0 to
15 output data line D0. This logic 1 is applied to incrementer
16 H1, which causes incrementer H1 to output the value 1 on the
17 five-bit bus leading to incrementer H2. (This value 1 will
18 prevent any of decoders DEC 0-1 through DEC 0-63 from
19 turning on corresponding transistors T0-1 through T0-63 to
20 place a conflicting value from bits B1 through B63 onto
21 output data line D0.)

22 If mask register bit M1 is logic 0, no value will be
23 added by incrementer H1 to the value 1 input to incrementer
24 H1. Thus the value 1 will be output by incrementer H1.
25 Even though decoder DEC 1-1 would decode the value 1, the
26 logic 0 value of M1 disables decoder DEC 1-1. Thus the
27 value B1 is not placed onto either of data lines D0 or D1.
28 If mask register bit M2 is logic 1, a 1 will be added by
29 incrementer H2 to the input value 1 and output to
30 incrementer H3 (not shown for clarity). Since M2 is logic
31 1, decoders in that row are enabled. Therefore, the value 1
32 input to incrementer H2 is decoded by decoder DEC 1-2, which
33 turns on transistor T1-2 to place the bit line signal B2
34 onto data line D1. From the above discussion, it can be
35 seen that other values in mask register M will produce other
36 connections from a bit line to an output data line.

37 Decode circuitry for mask bits M0 through M31 is as

1 discussed above. For mask register bits M32 through M63, no
2 more decoders are added, in the present embodiment, because
3 the data bus includes only 32 data lines D0 through D31. In
4 this portion of the circuit, error detection circuitry is
5 included comprising an OR gate which detects an overflow if
6 the number of logic 1's in mask register M is greater than
7 32. The error detection circuitry for a mask register bit
8 Mn is shown. OR gate ORn receives a logic 1 if incrementer
9 Hn detects that a 33rd logic 1 has been entered into mask
10 register M. At its other input, OR gate ORn receives a
11 logic 1 if any lower order incrementer has detected an
12 overflow. This logic 1 value propagates through all OR
13 gates above ORn and causes AND gate ANDn and all AND gates
14 above ANDn to output a logic 0, thus disabling all decoders
15 above row n.

16 Thus it can be seen that the circuit of Fig. 36a forms
17 a set of data bus outputs as specified by 64-bit mask
18 register M from the 64 bit line inputs B0 through B63, and
19 right-justifies the selected bits. Yet each selected bit
20 line value passes through only a single transistor to reach
21 its data line.

22 Decoders in Fig. 36a are preferably implemented as NOR
23 gates, though other implementations are of course possible.
24 The incrementer circuits of Fig. 36a may be implemented as
25 shown in the inset attached to incrementer H63.

26 Fig. 36b illustrates another shift and mask register
27 similar to that of Fig. 34a but having 16 data bits and an
28 8-bit data register. Mask 200 allows the shift and mask
29 circuit of Fig. 36b to select up to 8 of the 16 DATA IN bits
30 b₀ through b₁₅ to place on the DATA OUT bus. Values M₀
31 through M₁₅ are loaded into mask register 200. As with Fig.
32 34a, a value 1 in the mask register causes a bit value b₀
33 through b₁₅ to be shifted down and to the right, whereas a
34 value 0 in the mask register causes the bit value to be
35 shifted straight down. If mask register 200 contains more
36 than eight 1's, the higher order bit values will be lost.

37 Writing to the mask register is an inherently faster

1 operation than accessing configuration memory because it
2 does not involve setting up the long bit and word lines in
3 the configuration memory via the row and column decoders.
4 Thus, there is likely to be adequate time in a normal write
5 cycle to allow the decoding circuitry to settle. If there
6 are multiple mask registers selected by address bits, and
7 only a single set of decoding circuitry, then the decoding
8 delay will be incurred during the access to the user
9 register. Thus, the shift and mask unit of Fig. 36a is
10 mainly of benefit when there is only a single mask register.

11

12 Access to Registers Implemented Horizontally

13 The interfaces to registers and gate outputs provide
14 for word-wide access to registers in the device running in
15 the vertical direction so that the bits of the register all
16 occur on different bit lines. If the register runs
17 horizontally then all bits will occur in the same bit line
18 and parallel word-wide access will not be possible. Because
19 the number of bits of control store corresponding to state
20 access is likely to be approximately 20 times less than the
21 total number of bits of control store, it is quite feasible
22 to use "dual-ported" memory for the feedback bits along with
23 a second set of bit and word line drivers running in the
24 perpendicular direction to the first set. The extra port
25 allows horizontal as well as vertical registers to be
26 accessed in a word-wide fashion. Dual-ported memories are
27 well known and are disclosed in the above mentioned Weste &
28 Eshraghian book. This second set of drivers may have their
29 own dedicated shift and mask unit and wildcard register or
30 share with the first set according to detailed layout
31 considerations.

32

33 Control Store Duplication

34 Control store duplication will now be described. In
35 some situations it is convenient to have multiple bits of
36 the control store of an FPGA which are guaranteed to always
37 contain the same value. This duplication of control store

1 bits can eliminate wiring which would otherwise be required
2 to route the output of a single control store bit to a
3 distant location in the array. One important example of the
4 application of this technique is when a single routing wire
5 is provided which can be driven by more than one possible
6 source using 3-state drivers and the control store contains
7 bits which select which driver is active. A solution may be
8 achieved by routing the bits of the control store in
9 parallel with the wire itself to all drivers, but this
10 involves considerable area overhead for a long wire. An
11 alternative solution is achieved by simultaneously writing
12 duplicate bits to those control store locations which must
13 be identical. If the duplicate bits are on the same bit
14 line of the control store address, then simultaneous writing
15 of the duplicate bits is readily achieved by using the same
16 column address for the various columns of RAM containing
17 duplicate bits. By increasing the complexity of the row and
18 column decoders, for example by providing more than one NOR
19 gate in a given decoder and routing row address bits to the
20 column decoders and vice versa, a flexible structure can be
21 built which reads and writes the duplicate bits.

22 This arrangement is best seen in Fig. 38. The letter A
23 represents those memory cells (cells 351 and 352 are shown)
24 in which bits are to have the same value. Additional memory
25 cells A may be provided but are not shown in Fig. 38. All
26 memory cells A which are to have the same value are placed
27 on the same bit line 99. Other cells such as 331-334
28 labeled RAM are each separately addressable and are in the
29 same columns (word lines) as memory cells A. Word lines 361
30 and 362 can be selected by two different column addresses.
31 Word line 361 is selected by either of decoders 321 or 322
32 and word line 352 is selected by either of decoders 323 or
33 324. Decoders 321 and 323 decode the same address, and such
34 decoders are provided for all columns in which memory cell A
35 is located. In other words, all columns having a memory
36 cell A include a decoder which decodes a single row and
37 column address. Decoder 322 decodes the column address for

1 RAM cells 331 and 333, while decoder 324 decodes the column
2 addresses for RAM cells 332 and 334. Decoders 321 and 323
3 for memory cell A include row address bits for selecting bit
4 line 99, so the outputs of decoders 321 and 323 go high only
5 when the columns having memory cells A are selected and bit
6 line 99 is also selected. Decoders 322 and 324 for the RAM
7 cells go high only when bit line 99 is inactive. Thus,
8 multiple memory cells A can be simultaneously read or
9 written, and yet high density in the remainder of the memory
10 is retained. The remainder of the memory remains high
11 density because no extra word lines are added for accessing
12 the duplicate bits A. Another useful way of applying
13 control store duplication is to feed the read/write signal
14 to the address decoders and set decoders 321 to decode logic
15 0 and decoder 323 to decode logic 1 on the read/write line.
16 Feeding the read/write signal to other decoders such as 322
17 and 324 allows two row and column addresses, one for reading
18 and one for writing to a cell's function unit register to be
19 mapped onto a single address in the device address space.

20 The exact structure of the row and column decoders will
21 depend on a variety of factors, for example, the way in
22 which the duplicated bits are interspersed through the
23 control store and the performance required for read and
24 write operations. Appropriate circuit designs for decoders
25 can easily be arrived at using conventional design
26 techniques as disclosed in the aforementioned Weste &
27 Eshraghian book.

28

29 Processor Interface to FPGA

30 The processor interface will now be described. Current
31 FPGA designs do not provide any means for handshaking
32 information transfers between the user logic on the FPGA and
33 host microprocessors. Consequently, a variety of ad hoc
34 mechanisms have been used. The most flexible existing
35 mechanism presently in use is to clock the FPGA directly
36 from the processor, which keeps the two computations in
37 complete synchronisation. However, clocking the FPGA from

1 the processor slows the FPGA down too much in high
2 performance applications. Thus, clocked transfer is most
3 useful when relatively small amounts of data are transferred
4 to and from registers implemented on the FPGA and provides a
5 useful debugging methodology.

6 It is also possible to write data into a buffer memory
7 and then initiate a free-running clock implemented on the
8 FPGA itself or implemented on adjacent logic which runs for
9 enough cycles to complete the operation on the data and then
10 stops. This technique works efficiently for large data
11 streams but the overhead of initiating the clock in a
12 separate operation is significant for single operands
13 written to and read from the registers on the FPGA. The
14 processor can poll a hardware flag continuously, use an
15 interrupt generated by the FPGA or wait for a known delay
16 until the FPGA has finished computing and then reads back
17 the results. When an interrupt generated by the FPGA is to
18 be used by the processor, it may be convenient to provide a
19 small number of global output signals which can be
20 selectively driven by any cell function unit output. These
21 global signals can be used as interrupt request signals.
22 (The global signals may be implemented as wired ORs so that
23 several cells can activate the external interrupt. The FPGA
24 device may be programmed to latch the external interrupt
25 line until the latch is cleared by the processor writing to
26 an interrupt status register.

27 In many applications it is desirable to initiate
28 processing on the FPGA directly by the action of writing
29 data into registers on the array. We described above the
30 addressing scheme for input/output transfers from internal
31 device state registers where the row and column (bit and
32 word) wires used in this addressing scheme pass through the
33 array and have exactly the signals required to synchronise
34 computations on the FPGA. For example, in the case of a
35 write to a vertical register along the bit line, the word
36 line for those bits of RAM will go high during the transfer
37 and low when the transfer is complete. Although these bit

1 line and word line signals are normally concerned only with
2 programming and state access, they can easily be provided as
3 a source to one of the logic routing multiplexers in the
4 array. (Conveniently at a length-4 switch block, bit lines
5 are connected to East/West switches and word lines to
6 North/South switches.) Thus, user defined logic in the
7 FPGA can be triggered by the low going edge on the word line
8 signal to initiate the next computation to clock a new value
9 into the register.

10 When a relatively short operation (that is less than
11 the execution time of a small number of processing
12 instructions, say 500 nanoseconds with today's technology)
13 is implemented in the FPGA, it can be convenient to extend
14 the above state access mechanism by using the ability of
15 most processors to lengthen read and write cycles by
16 inserting "wait-states" when dealing with slow devices. The
17 FPGA generates a "wait" signal when the register is accessed
18 before its new value has been computed, forcing the
19 processor to wait until it can read the valid result.
20 Similarly, in the write cycle the processor is held up until
21 the previous data in the register has been processed. This
22 arrangement provides a very simple and low overhead method
23 of synchronising the processor with the FPGA.

24

25 CAD Software Tools

26 We will now describe and discuss the CAD software tools
27 for the FPGA and thereafter we will discuss the application
28 of CAL II to the implementation of several common logic
29 structures.

30 The present CAD tools for FPGAs represent a design as a
31 static hierarchy of connected components; this hierarchy is
32 then flattened into a set of basic components and converted
33 into a bit pattern representing the configuration of the
34 whole device. In a fine-grained FPGA, hierarchical blocks
35 of the design normally specify a rectangular area of the
36 fine-grained array and map onto a rectangular area in the
37 device configuration memory. If the FPGA is designed such

1 that the bit patterns for a given rectangular array of
2 memory depend only on the configuration of the resources in
3 the corresponding area of the design, it is possible to
4 program the FPGA rapidly on a block-by-block basis, all
5 instances of the same block having the same bit pattern.
6 Two instances of the same block with different external
7 connections may have different programming bit patterns.
8 With current FPGAs the configuration generation program
9 performs transformations on the user's design. The
10 transformations require the configuration program to be able
11 to analyse the entire design in order to determine the
12 configuration information for a block of that design.

13 One way for dynamic reconfiguration to be used is for
14 the host processor to construct the CAL II design
15 dynamically with an internal data structure and compute the
16 bit patterns corresponding to the design and then download
17 them directly into the chip. In such a case there is no
18 specialist translation program or static file containing
19 configuration bit patterns. This approach is made practical
20 by having a less highly encoded translation between design
21 representation and bit pattern (for example certain bits in
22 the bit pattern are reserved for representing a single
23 parameter). Translation can be applied hierarchically or on
24 a block-by-block basis. In addition, the fact that every
25 instance of the same block has the same configuration can be
26 used in conjunction with the multiple write capability of
27 the CAL II chip (implemented by wildcard registers) to
28 decrease programming time. The shift and mask register
29 feature allows overlapping blocks, each of which specifies
30 some resources in the same cell and the same word of control
31 memory, to be programmed independently by allowing a sub-set
32 of bits in a byte to be changed.

33

34 Easy Reconfiguration Through Block Design

35 Although an algorithm may be used to construct some CAL
36 II designs, in most cases users will wish to use more
37 traditional CAD tools to generate the CAL II designs without

1 losing the advantages of dynamic reprogramming. Dynamic
2 reprogrammability may be achieved by using replaceable
3 blocks. For each block of the design the user specifies a
4 number of possible configurations where each of these
5 configurations is a static design which can be produced and
6 analysed using conventional tools. Configuration data for
7 each potential configuration of each replaceable block and
8 the single initial configuration for the whole design can be
9 computed and stored in a disk file or non-volatile memory
10 such as an EPROM. A run-time library routine (that is, a
11 library routine written by the FPGA supplier and called by
12 the user's application programs to interact with the FPGA)
13 for the host processor which controls the CAL II chip can
14 then provide for replacing any replaceable block
15 configuration with one of its alternative configurations.
16 Replacement can be very simple and fast because it requires
17 only block transfers to regular areas of configuration
18 memory.

19 The software can also provide for initialisation of
20 state registers in replaceable blocks of the design.
21 Conveniently, state registers may be initialised to a
22 default value associated with the block definition or to the
23 previous state of the current instance of the block, thus
24 restoring its status. This can be achieved using the CAL II
25 architecture's ability to read and write registers randomly.

26 To ensure rapid reconfiguration, it is desirable to
27 impose some restrictions on replaceable blocks. For
28 example, each version of a replaceable block must have the
29 same bounding box, I/O signals must appear at the same point
30 on the block periphery on all versions of a replaceable
31 block, and no versions of a replaceable block may use any
32 chip resources which extend outside their bounding box. For
33 example, it would be unacceptable to use in a replaceable
34 block a flyover-wire which extended outside the bounding box
35 of the replaceable block. A more restrictive rule which
36 considerably simplifies the software is that no chip
37 resources lying within the boundary of an instance of a

1 replaceable block may be assigned to any other block in the
2 design. CAD software can easily check whether these
3 restrictions have been met. If they are not met, the block
4 can be ruled illegal. Alternatively, a more general purpose
5 and slower reconfiguration algorithm which checks individual
6 resources for conflicts rather than checking bounding boxes
7 can be used.

8 In some cases, there are relatively few potential
9 configurations of the device, and extremely rapid switching
10 between these configurations is desirable. In such a
11 situation, in order to minimise the number of device
12 accesses, optimisation software (which could have a long
13 run-time) may be used to analyse the device configuration
14 file and a list of potential reconfigurations. This
15 optimisation software will produce a set of configuration
16 operations which take advantage of the multiple write
17 capabilities of the device and change only those bits of
18 control store which are different. The optimiser output
19 could be stored in high level language code or machine
20 language program segments for a host processor. These pre-
21 computed instructions, when executed, will then perform the
22 reconfiguration rather than a data file controlling
23 reconfiguration.

24 Fig. 39 is a schematic diagram of an FPGA shown located
25 on an address bus and data bus with a microprocessor and
26 memory (EPROM and/or RAM). This depicts the simplicity of
27 using the FPGA in a microprocessor based circuit
28 application. The CAL II architecture does not support bi-
29 directional and tri-state wires. The principal reason for
30 this is that the CAL II is intended to support dynamic
31 reconfiguration by user software. It is possible that
32 during dynamic reconfiguration, the control store may be
33 configured incorrectly either as the result of the program
34 being terminated mid-way through configuring the array, or
35 because of errors in the user's software. In an
36 architecture where a wire can be driven by multiple
37 transceivers each of which is controlled by an independent

1 bit of RAM, there is the inherent potential for conflict,
 2 resulting in high power dissipation and potential damage to
 3 the device if the control store is incorrectly configured.
 4 Such a situation is tolerable when configurations are static
 5 and generated by trusted software, but is unacceptable in a
 6 device intended to support frequent reconfiguration. The
 7 function of tri-state buses can be emulated using wire-OR or
 8 wire-AND buses implemented using the cell logic gates and
 9 the longer logic wires provided by the CAL II array.

10

11 Example Applications Using CAL II

12 Figs. 40-48 show example applications of the CAL-II
 13 architecture. The drawing convention used in Figs. 40-48
 14 represents function units of a cell which are used by a
 15 design as a central box with a name on the box to represent
 16 the selected function. The drawing convention places
 17 signals in order according to decreasing length from the
 18 perimeter towards the center of the cell, so that, for
 19 example, length-4 flyovers near the perimeter of cells and
 20 the neighbour interconnects closer to the function block.
 21 Lines which turn at a single dotted line and pass to the
 22 central box in the cell represent signals being handled by
 23 one of multiplexers 58, 60, or 62 of Fig. 10. Lines which
 24 terminate at the edge of a box represent inputs to the
 25 function unit. The side of the function unit contacted
 26 corresponds to input terminals on Fig. 11 as indicated in
 27 Table I.

28

TABLE I

29	<u>Function Class</u>	<u>Left</u>	<u>Right</u>	<u>Top</u>	<u>Centre</u>
30	ZERO and ONE	Not used	Not used	Not used	F
31	A and \bar{A}	A	Not used	Not used	F
32	B and \bar{B}	Not used	B	Not used	F
33	Two Input Comb.	A	B	Not used	F
34	Multiplexer	A	B	Sel	F
35	Register	D	Clk	Clr	Q

36

37 Lines which exit from the center of a cell function unit

1 represent signals which have been placed on the SELF line by
2 function unit 48 of Fig. 10 and further connected to a
3 neighbour cell by one of multiplexers 50, 52, 54, or 56.
4 Lines which pass through one cell close to the function unit
5 and to the next cell represent signals being received on a
6 N, S, E, or W input by one of multiplexers 50, 52, 54, or 56
7 of Fig. 10 and passed to NOUT, SOUT, EOUT, or WOUT by that
8 multiplexer. In order to simplify the drawings, the switches
9 18 or 20 (switches are illustrated in Figs. 15, 16, and 17),
10 which are shown and labeled in Fig. 41, are not labeled in
11 Figs. 42-48. These switches are positioned between the
12 double lines which separate cell blocks, as shown in Fig.
13 41.

14 Fig. 40 depicts a first implementation of an
15 application using the CAL II architecture using a 4-input
16 AND gate. The 4-input AND gate is provided in a 4 x 4 block
17 of cells which typically implements additional functions,
18 although for clarity only those cells which implement the
19 AND gate are shown. Wide gates are found in many important
20 logic structures including address decoders and the AND and
21 OR planes of ROM/PLA/PAL type structures. It is essential
22 to be able to implement such wide gates efficiently in terms
23 of both speed and area. The CAL II architecture supports
24 the fast implementation of these wide gates by using a tree
25 of two-input, one-output logic cells 12. In the tree
26 structure shown in Fig. 40, the delay grows logarithmically
27 rather than linearly with the number of inputs. The drawing
28 convention in Figs. 40 through 48 represents function units
29 within the logic cells as rectangles labeled with their
30 selected function. Those cells whose function unit is
31 unused do not contain a rectangle. Input signals to the
32 logic cells are shown contacting the logic cell rectangles
33 at their edges, and outputs are shown leaving the logic cell
34 rectangles from their centers. Switches which connect
35 neighbor cells are positioned on the single dashed lines,
36 but for clarity are not shown. Switches 18, which were
37 illustrated in Figs. 1, 2, 15, and 16 are also not shown but

1 are positioned between the double dashed lines.

2 In Fig. 40, AND gate 12a receives two inputs, IN0 and
3 IN1, and AND gate 12c receives two inputs IN2 and IN3. The
4 outputs of AND gates 12a and 12c form the inputs to AND gate
5 12b from whence the output OUT is taken. The function units
6 depicted in Figs. 18 to 25 allow true or complemented values
7 of each input variable to be used, which is essential for
8 decoders. The flexibility of the function unit in the OR
9 plane of a ROM enables the number of product terms to be
10 halved, and the routing resources provided by the CAL II
11 architecture allow tree-structured gates with up to 32
12 inputs to be implemented in a single column of cells.

13 Fig. 41 depicts a 16-cell AND gate with the 16 cells
14 arranged in a column of the array. There are four 4-cell x
15 4-cell blocks arranged vertically. This arrangement not
16 only depicts the connections between neighbour cells but
17 also the connections between the blocks of cells using the
18 length-4 and length-16 flyovers. The cells are numbered
19 from cell 0 at the bottom to cell 15 at the top. In Fig.
20 41, switches 18 are shown located in the spaces between the
21 cell blocks. In addition, it will be seen that lines 210,
22 211 and 212 depict length-4 flyover routing. Signals can
23 only enter flyover 210 at switch 18 between cells 3 and 4
24 although a signal can exit flyover 210 directly into cells
25 4, 5, 6, 7 and into the switch 18 between cells 7 and 8. In
26 the bottom block comprising cell 0 through cell 3, there are
27 three AND gates in cells 0 to 2. The output of cell 1
28 passed through an unused cell 3 and enters the switch 18 at
29 the boundary to the bottom block. The output of cell 1
30 forms an input of the AND gate in cell 4. The output of the
31 AND gate in cell 4 is sent via the switch 18 between cells 3
32 and 4 to length-4 flyover 210. Cell 8 has one input from
33 length-4 flyover 210 and the other input from cell 12 via
34 flyover 211 (without going through the switch 18 between
35 cells 7 and 8). Output of the 16-input AND gate is taken
36 from the output of cell 8 and is routed via the switch 18
37 between cells 7 and 8 onto flyover 212 to the switch 18

1 between cells 11 and 12, and provided as output at the top
2 of Fig. 41.

3 Fig. 42 depicts a PAL-type structure showing how an AND
4 plane is built up and mated to an OR plane to form a general
5 purpose logic block. Inputs IN0 through IN15 are provided
6 to 8 columns of 16 rows of AND gates. Each column is
7 connected as shown in Fig. 41 to form a tree structure.
8 Because there are eight columns of AND gates, connections
9 from the input signals IN0 through IN15 are applied to
10 length-4 flyovers. Input signals are applied at the left of
11 the figure and the East flyovers are used. Since two input
12 signals are applied to AND gates at the lowest level of the
13 tree, and only one East length-4 flyover is provided for
14 each cell, in even rows, the East length-4 flyovers of
15 adjacent cells are used, and signals transferred through
16 neighbor interconnect. For example, the row of cells
17 labeled ROW 5 receives its IN5 input from the East length-4
18 flyovers of ROW 5 and its IN4 input from the East length-4
19 flyover of ROW 4. But in the embodiment shown in Fig. 10,
20 there is no provision to take routing from one neighbour
21 cell to another from a length-4 flyover. Therefore at the
22 switches indicated by double dotted lines the signal on the
23 east length-4 flyover is transferred to east neighbour
24 routing. One such switch transfer is labeled 424 in Fig.
25 42. For simplicity, other transfers are not labeled. The
26 IN4 signal is then transferred east through neighbour
27 routing to the next three adjacent cells. The signal is
28 also transferred by cells in row 4 upward along neighbor
29 interconnect to cells directly above in row 5. In
30 accordance with the invention, the IN4 signal runs on the
31 east length-4 flyover as well as through neighbour
32 interconnect, so that it reaches the four AND gates at the
33 right of the figure with less delay than if it had passed
34 through eight neighbour cells. Another switch equivalent to
35 424 transfers the IN4 signal to the neighbour interconnect
36 of the right 4 columns. In this application, it is not
37 necessary to also place the signal on the east length-4

1 flyover at the right of the figure because no further
2 connection of the IN4 signal to the right of the figure is
3 made. Rows 6, 8, 10, 12, and 14 include the same
4 combination of length-4 flyovers and neighbour routing to
5 get high speed. Rows 1 and 3 also include this combination,
6 though in these cases, the signal is passed downward to rows
7 0 and 2, respectively, rather than upward.

8 One row of 7 OR gates is positioned at the top of Fig.
9 42. OR gate OR0 receives as input the outputs of the first
10 two columns of AND gates. OR gate OR2 receives as input the
11 outputs of the third and fourth columns of AND gates. OR
12 gate OR1 receives as input the outputs of OR gates OR0 and
13 OR2. A similar tree structure is formed by OR gates OR5
14 through OR7, with the output signal OUT taken from OR gate
15 OR3 through a length-4 flyover.

16 Fig. 43 depicts a one-bit accumulator constructed from
17 a row of 5 cells in two (4 x 4) cell blocks. The cells are
18 configured for XOR, AND, MUX and DC as shown to create a SUM
19 output and a CARRY output.

20 Fig. 44 depicts a three bit accumulator with a look
21 ahead carry (for 3 inputs In0, In1 and In2 generating SUM0,
22 SUM1, and SUM2, and CARRYOUT).

23 Fig. 45 depicts an adder which is a 16 bit accumulator
24 with a look ahead carry for minimising the delay along the
25 carry chain. The CAL II architecture supports the 2:1
26 multiplexer as a cell function, as can be seen from Figs.
27 43, 44 and 45, and this reduces the carry path delay from
28 two gate delays in the CAL I architecture to one gate delay
29 in this architecture. The extra routing resources provided
30 by the flyovers allows the one bit adder shown in Fig. 43 to
31 be implemented in a single row of cells, which reduces
32 routing delays on the carry path as compared to a two-cell-
33 high (CAL I) implementation. In this way it will be
34 appreciated that accumulators and adders of various
35 complexity can be constructed using the CAL II architecture
36 and, of course, the routing resources can be used as shown
37 in Fig. 45 to route the carry from a previous stage over a

1 block of adders in a look ahead structure.

2 The CAL II architecture can also implement synchronous
3 counters. As described above, the CAL II architecture
4 provides an edge triggered flip-flop as a cell primitive,
5 which allows a more efficient implementation of such
6 synchronous counters. Fig. 46 depicts a 4-bit synchronous
7 counter stage which has the usual signals such as Clock
8 Enable In, Clock Enable Out, clock, and output signals Q0,
9 Q1, Q2 and Q3. It will be seen that the clock enable output
10 signal comes from a length-4 flyover and that the clock
11 signal can be communicated to all cells in the row used for
12 flip flops via the length-4 flyover. The CAL II
13 architecture is particularly effective because it provides
14 flyover routing resources to route the clock lines directly
15 into the cells. Also, the look ahead function required by
16 the fast synchronous counters is provided using wide gates.
17 The 4-bit counter stage shown in Fig. 46 can be cascaded and
18 expanded to form a 16 bit synchronous counter as shown in
19 Fig. 47 using 4 blocks of 4 cells x 4 cells.

20 It will also be appreciated that wide multiplexers such
21 as 16:1 multiplexers can be efficiently implemented as a
22 tree of 2:1 multiplexers. Such an arrangement is depicted
23 in Fig. 48 in which two 4 x 4 blocks of cells are used to
24 form the tree. The first row of cells has eight cells
25 implementing 2:1 multiplexers. The outputs of these
26 multiplexers are fed to the inputs of multiplexer cells in
27 the second row whereupon two outputs are taken from the 2:1
28 multiplexer shown in the third row which provides the output
29 of the 16:1 multiplexer.

30

31 Summary

32 It will readily be appreciated that all common logic
33 structures can be implemented using this technology. The
34 main additional features supported by the CAL II
35 architecture are that the control store layout is arranged
36 so that closely associated groups of resources within a cell
37 are accessed through the same byte of the control store,

1 additional logic circuits on the control store interface
2 allow for word-wide read and write access to internal state
3 registers in the user design, wildcard registers are
4 provided in the control store address decoder to allow
5 vectors of cells and bit slices in the user designs to be
6 changed simultaneously, and a hierarchical routing structure
7 consisting of length-4, 16 and 64 wires is overlaid on the
8 basic cell grid to reduce the delay on longer wires. As
9 described above, length-4 wires are used as function unit
10 inputs to the basic cells. This structure can be extended
11 upwardly in a hierarchical manner to length-64 and
12 length-256 and so on for wires in future product families.

13 The CAL II architecture provides the ability to make
14 dynamic accesses to a CAL II FPGA by mapping its control
15 store into the address space of a host processor. This
16 offers design opportunities which are not available from any
17 other FPGA. Significant benefits can be gained from this
18 architecture without abandoning traditional CAD frameworks.
19 The CAL II architecture can be used in a variety of modes
20 and four principal modes have been identified:

21 1. Conventional ASIC: In this mode, conventional
22 ASIC/FPGA design tools are used to produce a static
23 configuration file which is then loaded into the device from
24 an EPROM or other non-volatile store at power up. No host
25 processor is needed, although it will be appreciated that if
26 such a host processor is available, savings in board area
27 can be obtained by storing the CAL II design configuration
28 within the host processor's memory. The use of the host
29 processor also allows configuration time and configuration
30 data size to be greatly reduced by taking advantage of the
31 wildcard units in the CAL II address decoders.

32 2. Processor Access to Internal State: In this
33 arrangement, again a conventional ASIC process flow is used
34 to produce a static configuration which is then down-loaded
35 on power up. While the device is active the processor
36 accesses internal registers of the user's design to store
37 and to retrieve values. The control store interface can be

1 regarded as providing free wiring to all internal registers
2 of the user's design. Use of existing control store wiring
3 can increase density by eliminating wires which would
4 otherwise be required to route signals to the chip edge, and
5 can also reduce design complexity. This design style is
6 particularly attractive in applications where the FPGA
7 provides an interface between a microprocessor and external
8 hardware. Software running in the host calculates the
9 addresses of internal registers using trivial computations
10 based on placement information output from the CAD system.

11 3. Multiple Unrelated FPGA Configurations: In this
12 design style several complete FPGA designs are undertaken in
13 parallel using a conventional CAD system and then verified
14 independently. Run-time software on the host processor can
15 then swap between various configurations of the FPGA device.
16 Conveniently, FPGA configurations can be associated with
17 processes running on the host processor and swapped during
18 process context switches, preserving the state of internal
19 registers. In this way, each process can appear to have
20 access to its own 'virtual' FPGA. These multiple
21 configurations must, however, be designed to co-operate with
22 each other if any user I/O pins are shared by multiple
23 configurations. The additional circuits on the CAL II
24 control store interface greatly reduce the number of write
25 operations to switch between various device configurations.
26 One example of an application suited to this technique is a
27 laser printer controller where the FPGA initially operates
28 as an RS232 interface to down-load a printer image file and
29 is then reconfigured to control the print engine and
30 implement low level graphics operations.

31 4. Algorithmic Use of Dynamic Reconfiguration: In
32 this design style portions of the circuit implemented on the
33 FPGA are reconfigured dynamically as part of the computation
34 being performed. For example, the routing network in the
35 FPGA may be used directly to perform a permutation function
36 over the FPGA input pins. The largest part of the design
37 work and much of the verification can be done using

1 conventional ASIC design tools.

2 A high percentage of system designs in present use
3 consist of a processor, memory, and chips to interface to
4 I/O devices on the circuit board. The design of such a
5 system consists of both hardware design of any ASIC or FPGA
6 and the board itself, and also software design of the
7 program for the processor which implements most of the
8 desired functionality. Mapping of the control store of the
9 FPGA into the address space of the processor provides the
10 opportunity to move elements of the design from the hardware
11 engineer to the software engineer which simplifies the
12 overall design process. It is still necessary for software
13 to lay out the user's design onto the hardware of the CAL II
14 device, but the software for this task can be less complex
15 because of the regularity of the CAL II architecture.

16 A principal advantage of the CAL II structure is that
17 it is simple, symmetrical, and regular, which allows novice
18 users to quickly make use of the array of fine-grained
19 cells, and permits CAD tools to make efficient use of the
20 resources available. A further advantage of the CAL II
21 array is that it provides flexibility in placing functional
22 blocks of designs on the array to meet an overall size
23 constraint. The arrangement of the control store and the
24 use of the wildcard registers and shift and mask registers
25 minimises the number of microprocessor instructions required
26 to access device resources and status. The specific
27 structure of the control store allows many control bits to
28 be written simultaneously instead of one at a time because
29 of the structured set of data in the RAM. This has the
30 advantage of reducing the testing overhead because testing
31 uses regular configurations. The advantage of the
32 hierarchical scaling is that delays are logarithmic in terms
33 of distance in cell units and delays are hence considerably
34 reduced in comparison with previous designs. Since the
35 flyover wires can only be driven by one element, dynamic
36 access to the control store is safer because there is no
37 possibility of incorrect configurations causing contention.

1 This added safety is useful in situations where the FPGA
2 configuration is intended to be frequently altered by a
3 user.
4

1 CLAIMS

2 I claim:

3 1. An hierarchically-scalable cellular array
4 comprising:

5 a plurality of cells;

6 a first routing resource for interconnecting each cell
7 to its adjacent neighboring cells, said first routing
8 resource including a first plurality of lines having a
9 length n , where n is a predetermined distance;10 a second routing resource including a second plurality
11 of lines having a length $N \times n$ where $N > 1$; and12 a third routing resource including a third plurality of
13 lines having a length $N^m \times n$ where $m \geq 2$ and where $N^m \times n \leq T$,
14 the total length of said cellular array.

15

16 2. The cellular array of Claim 1 wherein n is
17 approximately one cell width.

18

19 3. The cellular array of Claim 1 wherein N is 4.

20

21 4. The cellular array of Claim 3 wherein m is 2.

22

23 5. The cellular array of Claim 3 wherein m is 3.

24

25 6. The cellular array of Claim 1 further including a
26 fourth routing resource including a fourth plurality of
27 lines having a length $N^r \times n$ where $r \geq 3$, and $r = m + 1$.

28

29 7. A programmable logic structure comprising:

30 an array of cells arranged in rows and columns;

31 a plurality of switches, said plurality of switches
32 partitioning said array into a first plurality of cell
33 blocks,

34 wherein each cell includes:

35 four input lines for providing four input
36 signals to said cell, each of said input lines
37 coupled to an adjacent cell or switch; and

1 four output lines for providing four output
2 signals from said cell, each of said output lines
3 coupled to an adjacent cell or switch,
4 wherein said array further includes four intermediate,
5 directed input lines for providing an additional four input
6 signals to said cell, wherein each intermediate input line
7 provides signals to the cells in either a row or a column of
8 one of said blocks.

9
10 8. The logic structure of Claim 7 wherein said array
11 further includes a plurality of flyover lines, each flyover
12 line coupled between two of said plurality of switches,
13 wherein said plurality of flyover lines determines a second
14 plurality of cell blocks.

15
16 9. In an integrated circuit having a two-dimensional
17 array of logic cells, a switch comprising:
18 a set of multiplexers, each of which receives a
19 plurality of input signals and provides an output signal;
20 for each multiplexer, multiplexer control means for
21 causing said multiplexer to select one of said input signals
22 to provide said output signal, said input signals being
23 taken from the following wires:

24 a wire extending to the west from said switch and
25 having a length approximately equal to a first multiple
26 of said length of one of said cells;

27 a wire extending to the east from said switch and
28 having a length approximately equal to said first
29 multiple of said length of one of said cells;

30 a wire extending to the west from said switch and
31 having a length approximately equal to a second
32 multiple of said length of one of said cells; and

33 a wire extending to the east from said switch and
34 having a length approximately equal to said second
35 multiple of said length of one of said cells;

36 said output signals being placed on the following wires:

37 a wire extending to the west from said switch and

1 having a length approximately equal to a first multiple
2 of said length of one of said cells;

3 a wire extending to the east from said switch and
4 having a length approximately equal to said first
5 multiple of said length of one of said cells;

6 a wire extending to the west from said switch and
7 having a length approximately equal to a second
8 multiple of said length of one of said cells; and

9 a wire extending to the east from said switch and
10 having a length approximately equal to said second
11 multiple of said length of one of said cells.

12

13 10. The switch of Claim 9 wherein said first multiple
14 is 1 and second multiple is 4.

15

16 11. The switch of Claim 9 wherein said first multiple
17 is 4 and second multiple is 16.

18

19 12. The switch of Claim 9 wherein said first multiple
20 is 16 and second multiple is 64.

21

22 13. The switch of Claim 9 wherein said
23 multiplexers further receive input signals taken from the
24 following wires:

25 a wire extending to the west from said switch and
26 having a length approximately equal to a third multiple of
27 said length of one of said cells; and

28 a wire extending to the east from said switch and
29 having a length approximately equal to said third multiple
30 of said length of one of said cells.

31

32 14. The switch of Claim 13 wherein said first multiple
33 is 1, said second multiple is 4, and said third multiple is
34 16.

35

36 15. The switch of Claim 13 wherein said first multiple
37 is 4, said second multiple is 16, and said third multiple is

1 64.

2

3 16. A programmable integrated circuit comprising:

4 a two-dimensional array of logic cells, each cell
5 comprising means for generating as an output signal a
6 selected logic function of a plurality of logic cell input
7 signals;

8 a plurality of switches, said switches positioned so as
9 to group said cells into cell blocks;

10 each cell including:

11 four short input wires for providing
12 four of said input signals, said four short
13 input wires carrying output signals from the
14 cells or the switches which comprise nearest
15 neighbors on four sides of said cell;

16 four medium input wires for providing
17 four of said input signals, said four medium
18 input wires extending in four compass
19 directions between switches which define a
20 cell block in which said cell is positioned,
21 wherein a plurality of said medium input
22 wires are simultaneously accessible by said
23 cell; and

24 four short output wires for providing
25 said output signal, said four short output
26 wires carrying output signals to the cells or
27 the switches which comprise nearest neighbors
28 on four sides of said cell.

29

30 17. The programmable integrated circuit of Claim 16
31 wherein each of said medium input wires carries a signal
32 from a switch in one of the four compass directions to said
33 cell and to other cells positioned between two of said
34 switches making up said cell block.

35

36 18. The programmable integrated circuit of Claim 17
37 wherein said medium wires are approximately four times the

1 length of said short wires.

2

3 19. A programmable logic structure comprising:

4 an array of cells;

5 a plurality of switches arranged to group said cells
6 into cell blocks such that for each cell block at least one
7 west switch is positioned to the west of that block, and at
8 least one east switch is positioned to the east of that
9 block;

10 a neighbor wire connecting from an output of said west
11 switch to an input of the nearest cell east of said west
12 switch;

13 a flyover wire connecting from an output of said west
14 switch to an input of said east switch;

15 a neighbor wire connecting from an output of said east
16 switch to an input of the nearest cell west of said east
17 switch; and

18 a flyover wire connecting from an output of said east
19 switch to an input of said west switch.

20

21 20. A programmable logic structure of Claim 19 wherein
22 said plurality of switches is further arranged such that for
23 each cell block at least one north switch is positioned to
24 the north of that block, and at least one south switch is
25 positioned to the south of that block, wherein said
26 programmable logic structure further comprises:

27 a neighbor wire connecting from an output of said
28 north switch to an input of the nearest cell south of
29 said north switch;

30 a south flyover wire connecting from an output of
31 said north switch to an input of said south switch;

32 a short wire connecting from an output of said
33 south switch to an input of the nearest cell north of
34 said south switch; and

35 a north flyover wire connecting from an output of
36 said south switch to an input of said north switch.

37

1 21. A programmable logic structure as in Claim 20 in
2 which each of said cells comprises a plurality of smaller
3 cells, each of said smaller cells being defined by:
4 a plurality of switches, said switches arranged to
5 group said smaller cells into smaller cell blocks such
6 that for each smaller cell block at least one west
7 switch is positioned to the west of that block, at
8 least one east switch is positioned to the east of that
9 block, at least one north switch is positioned to the
10 north of that block, and at least one south switch is
11 positioned to the south of that block;
12 said smaller cells being connected to said switches by:
13 a neighbor wire connecting an output of said west
14 switch to an input of the nearest smaller cell east of
15 said west switch;
16 a neighbor wire connecting an output of the
17 nearest smaller cell east of said west switch to an
18 input of said west switch;
19 an east flyover wire connecting from an output of
20 said west switch to an input of said east switch;
21 a neighbor wire connecting from an output of said
22 east switch to an input of the nearest smaller cell
23 west of said east switch; and
24 a neighbor wire connecting an output of the
25 nearest smaller cell west of said east switch to an
26 input of said east switch;
27 a west flyover wire connecting from an output of
28 said east switch to an input of said west switch.
29 a neighbor wire connecting from an output of said
30 north switch to an input of the nearest smaller cell
31 south of said north switch;
32 a neighbor wire connecting an output of the
33 nearest smaller cell south of said north switch to an
34 input of said north switch;
35 a south flyover wire connecting from an output of
36 said north switch to an input of said south switch;
37 a neighbor wire connecting from an output of said

1 south switch to an input of the nearest smaller cell
2 north of said south switch; and

3 a neighbor wire connecting an output of the
4 nearest smaller cell north of said south switch to an
5 input of said south switch;

6 a north flyover wire connecting from an output of
7 said south switch to an input of said north switch.

8

9 22. An input/output structure for a programmable logic
10 device having a plurality of logic cells comprising:

11 a plurality of pads, each pad connected to an external
12 pin of said programmable logic device;

13 a plurality of I/O buffers, each I/O buffer connected
14 to one of said plurality of pads, each I/O buffer
15 including a switch for connecting said I/O buffer
16 to at least one of said plurality of logic cells,
17 wherein said switch forms part of a hierarchial
18 interconnect structure; and

19 means for controlling said switch.

20

21 23. A method for routing signals in a programmable
22 logic structure comprising the steps of

23 arranging an array of cells in rows and columns;

24 partitioning said array into a plurality of cell blocks
25 with a plurality of switches;

26 coupling each input line associated with each cell to
27 an adjacent cell or switch;

28 coupling each output line associated with each cell to
29 said adjacent cell or switch;

30 providing an additional four directed, intermediate
31 input lines for each cell, wherein each directed
32 intermediate input line provides signals to the cells in
33 either a row or a column of one of said plurality of cell
34 blocks.

35

36 24. The method of Claim 23 further including the step
37 of

1 providing a plurality of flyover lines, each flyover line
2 coupling two of said plurality of switches.

3

4 25. The method of Claim 24 further including the step
5 of providing a signal from a first cell to a second cell,
6 wherein said providing includes transferring said signal via
7 at least one intermediate input line.

8

9 26. The method of Claim 25 wherein said step of
10 providing further includes transferring said signal via at
11 least one flyover line.

12

13 27. The method of Claim 23 further including the steps
14 of:

15 sending a signal via a first path including a plurality
16 of switching units, said first path paralleling a direct,
17 second path;

18 using logic gates to detect whether said signal travels
19 the full length of said first path; and

20 if said signal travels said full length, then placing
21 said signal on said second path.

22

23 28. A logic structure comprising:

24 an array of memory cells arranged in rows and columns;

25 a plurality of decoders, one decoder for each row or
26 each column, each decoder receiving both true and
27 complemented values for each address bit;

28 a pair of logic gates associated with each address bit,
29 wherein said pair of logic gates provide said true and
30 complemented values for each address bit; and

31 a wildcard register providing one enable bit for each
32 address bit, wherein both of said logic gates receive said
33 enable bit from said wildcard register, one of said logic
34 gates receives said address bit, and the other of said logic
35 gates receives the complement of said address bit.

36

37 29. A method of simultaneously writing a plurality of

1 cells comprising the steps of:
2 arranging a plurality of memory cells in rows and
3 columns;
4 providing a decoder for each row or each column, each
5 decoder receiving both true and complemented values for each
6 address bit;
7 providing a pair of logic gates for each address bit,
8 wherein said pair of logic gates provide said true and
9 complemented values for each address bit; and
10 placing a wildcard register in operative relation to
11 said pair of logic gates, wherein said wildcard register
12 provides one enable bit for each address bit, further
13 wherein both of said logic gates receive said enable bit
14 from said wildcard register, one of said logic gates
15 receives said address bit, and the other of said logic gates
16 receives the complement of said address bit.

17

18 30. A programmable logic device comprising:
19 an array of cells arranged in rows and columns;
20 a plurality of match registers, each match register
21 placed in operative relation to a row or column, wherein if
22 an address matches the stored value in a match register,
23 then said match register activates said row or column.

24

25 31. The programmable logic device of Claim 30 further
26 including a bit select register to select at least one
27 predetermined bit from said row or column.

28

29 32. A method of facilitating multiple writes
30 comprising the steps of:
31 arranging a plurality of cells in rows and columns;
32 placing a match register in operative relation to each
33 row or column;
34 comparing an address to a stored value in each match
35 register; and
36 activating said row or said column if said address
37 matches said stored value.

1
2 33. A register structure comprising:
3 a plurality of input lines;
4 a plurality of output lines;
5 a register providing a plurality of bits, the number of
6 bits equal to the number of said plurality of input lines;
7 a plurality of tiered switches placed in operative
8 relation to said input lines and said register, wherein said
9 plurality of output lines is coupled to said plurality of
10 input lines in a pattern determined by a value in said
11 register.

12
13 34. The register structure of Claim 33 wherein each
14 switch comprises:

15 a first transistor and a second transistor, each
16 transistor having an enable terminal, a first terminal
17 and a second terminal;

18 an enable line coupled to said enable terminal of
19 said first transistor and to an inverter which in turn
20 is coupled to said enable terminal of said second
21 transistor;

22 a first input line coupled to said first terminal
23 of said first transistor and a second input line
24 coupled to said first terminal of said second
25 transistor; and

26 an output line coupled to the second terminals of
27 said first and second transistors.

28
29 35. The register structure of Claim 34
30 wherein said first terminals of said switches on
31 said first input line are coupled to a voltage source,
32 wherein each bit of said register is provided to
33 those enable terminals associated with one tier of said
34 plurality of tiered switches, and further
35 wherein if one tier is not the last tier, then
36 said output line of one switch in said
37 one tier is coupled to the second terminal of

1 the switch in the next tier in the same input
2 line as said one switch, if present, and
3 said output line of said one switch in
4 said one tier is further coupled to the first
5 terminal of the switch in the next tier in
6 the data line adjacent said data line of said
7 one switch.

8
9 36. A method of providing access to a subset of a
10 plurality of bits of a configuration memory comprising the
11 steps of:
12 placing a register in operative relation to a plurality
13 of input lines, wherein said register provides one bit for
14 every input line;
15 positioning a plurality of switches in tiers in
16 operative relation to said input lines; and
17 interconnecting said plurality of output lines to said
18 plurality of input lines in a pattern determined by a value
19 in said register.

20
21 37. The method of Claim 36 wherein each switch
22 comprises a first transistor and a second transistor, each
23 transistor having an enable terminal, a first terminal and a
24 second terminal.

25
26 38. The method of Claim 37 wherein said step of
27 interconnecting includes the steps of:
28 coupling an enable line to said gate of said first
29 transistor and to an inverter which in turn is coupled to
30 said gate of said second transistor;
31 coupling a first input line to said first terminal of
32 said first transistor and coupling a second input line to
33 said first terminal of said second transistor;
34 coupling an output line to the second terminals of said
35 first and second transistors;
36 coupling said first terminals of said switches on said
37 first data line to a voltage source;

1 providing each bit of said register to said enable
2 terminals associated with one tier of said plurality of
3 tiered switches,
4 wherein if one tier is not the last tier, then
5 coupling said output line of one switch in said one tier to
6 the second terminal of the switch in the next tier in the
7 same input line as said one switch, if present, and coupling
8 said output line of said one switch in said one tier to the
9 first terminal of the switch in the next tier in the input
10 line adjacent said input line of said one switch.

11

12 39. A decoding system in an array of cells comprising:
13 a plurality of bit lines placed in operative relation
14 to said array;

15 a plurality of word lines placed in operative relation
16 to said array;

17 a plurality of address decoders, wherein each address
18 decoder is coupled to a word line; and

19 a plurality of duplicate decoders, wherein each
20 duplicate decoder is associated with a predetermined word or
21 bit line, said duplicate decoder and the corresponding
22 address decoder coupled to said predetermined word line via
23 a logic gate.

24

25 40. A method of manipulating a first set of bits to
26 produce a second set of bits comprising the steps of:

27 selecting a predetermined pattern which relates said
28 first set of bits to said second set of bits; and

29 configuring a register device to provide said pattern.

30

31 41. A routing device in a programmable logic device
32 comprising:

33 a plurality of cells providing a first path, wherein
34 each cell includes:

35 means for receiving a plurality of input
36 signals and providing an output signal;

37 a plurality of memory bits associated with

1 said means for receiving; and
2 a first logic gate for providing a trigger
3 signal determined by the state of said plurality
4 of memory bits;
5 a second path not including cells, said second path
6 paralleling said first path;
7 a second logic gate coupled to the output terminals of
8 the first logic gates to detect whether said signal travels
9 the full length of said first path; and
10 means for determining whether a signal is provided on
11 said first path or on said second path, said means for
12 determining controlled by said second logic gate.
13
14 42. A function unit comprising:
15 a plurality of multiplexers; and
16 at least one flip-flop,
17 wherein a first set of said plurality of multiplexers
18 receive input signals from a hierarchical interconnect
19 system, and a second second set of said plurality of
20 multiplexers receive output signals and their complements
21 from said first set of multiplexers,
22 wherein one of said plurality of multiplexers is a
23 function multiplexer which is controlled by an output signal
24 from one of the multiplexers in said second set, wherein
25 said function multiplexer receives output signals from the
26 other multiplexers in said second set,
27 wherein said second set of multiplexers provide input
28 signals to said at least one flip-flop, and
29 wherein one of said plurality of multiplexers receives
30 output signals from said function multiplexer and said at
31 least one flip-flop, and provides an output signal for said
32 function unit.

1/42

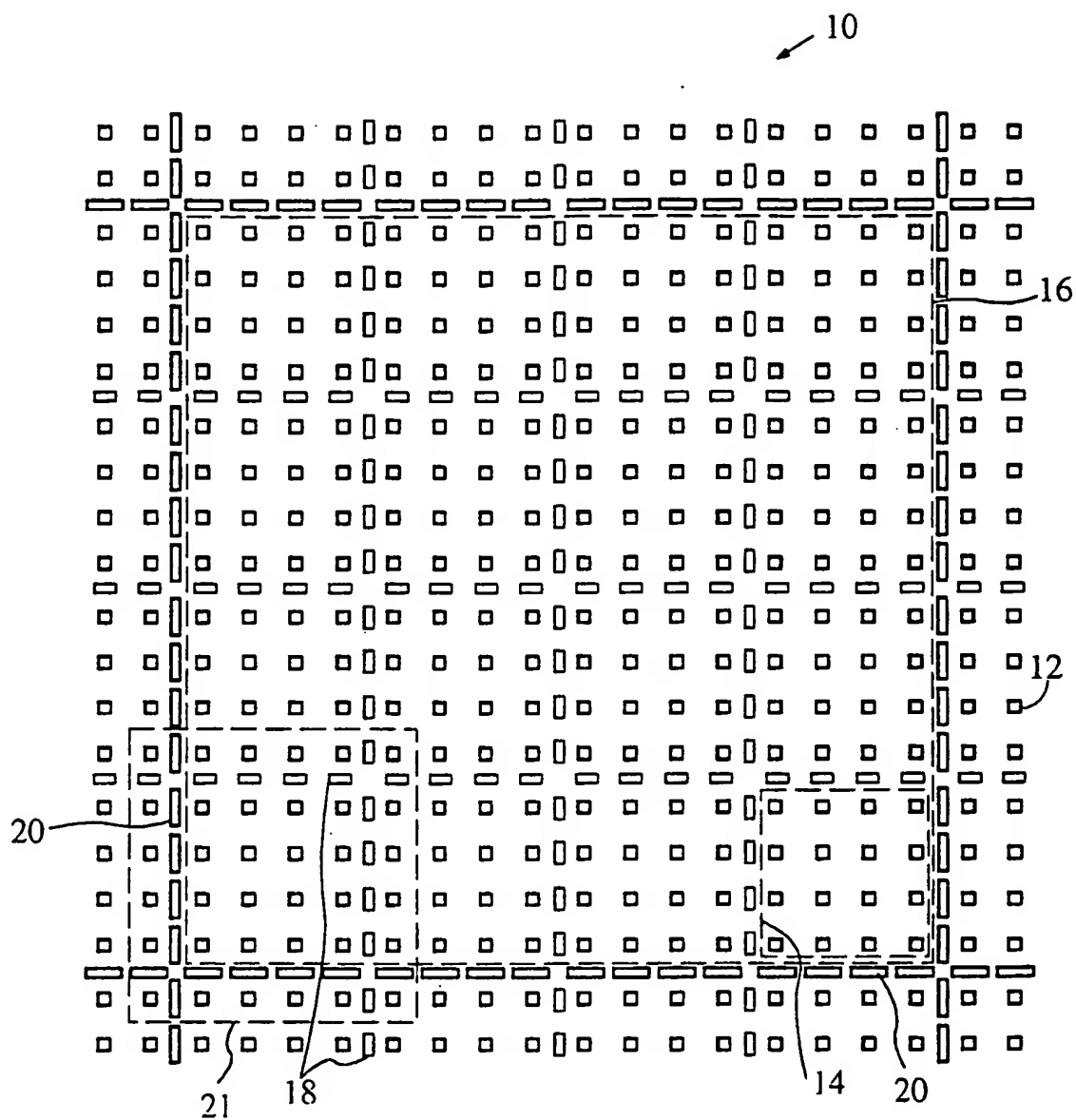


FIG. 1

2/42

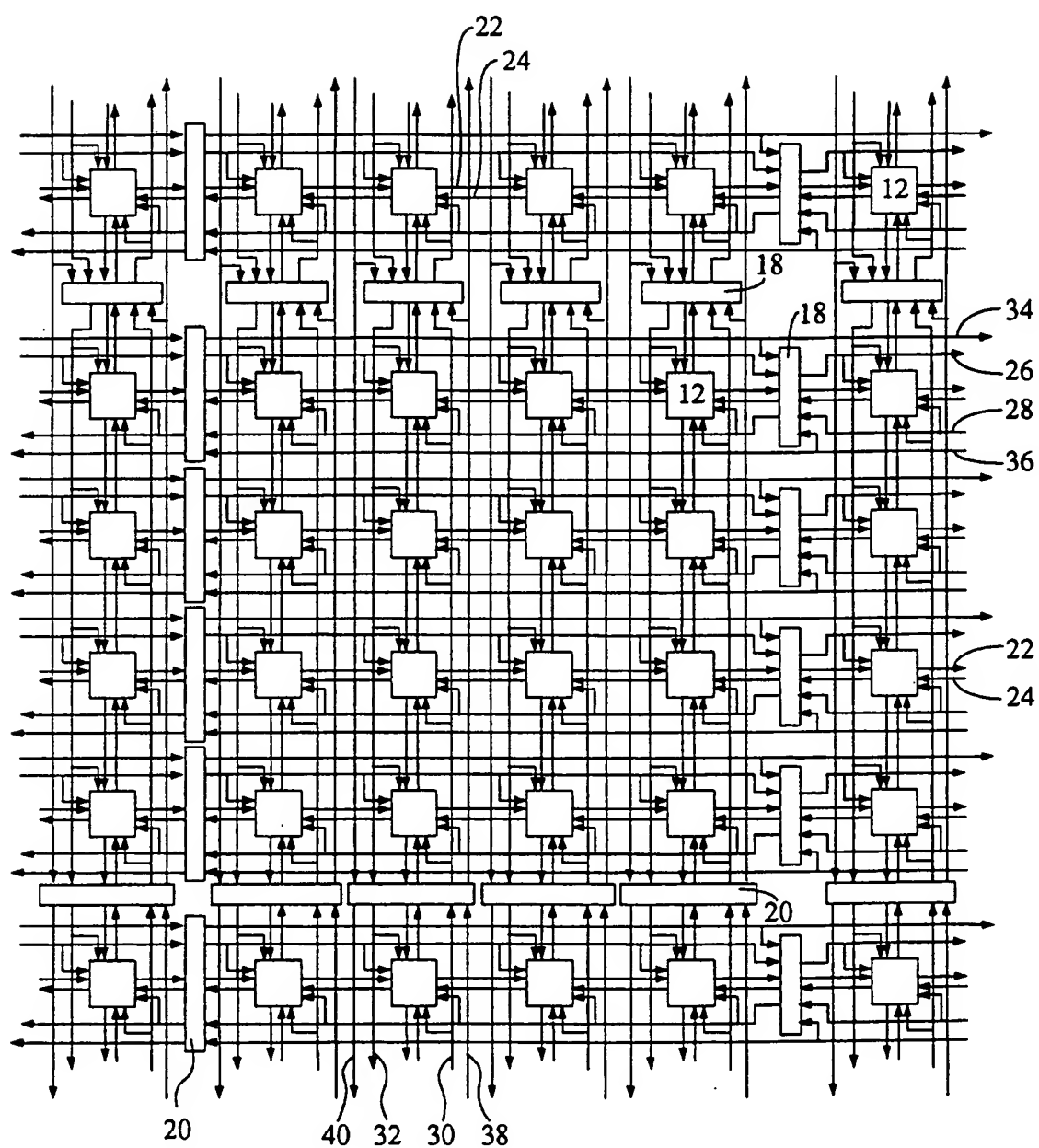


FIG. 2

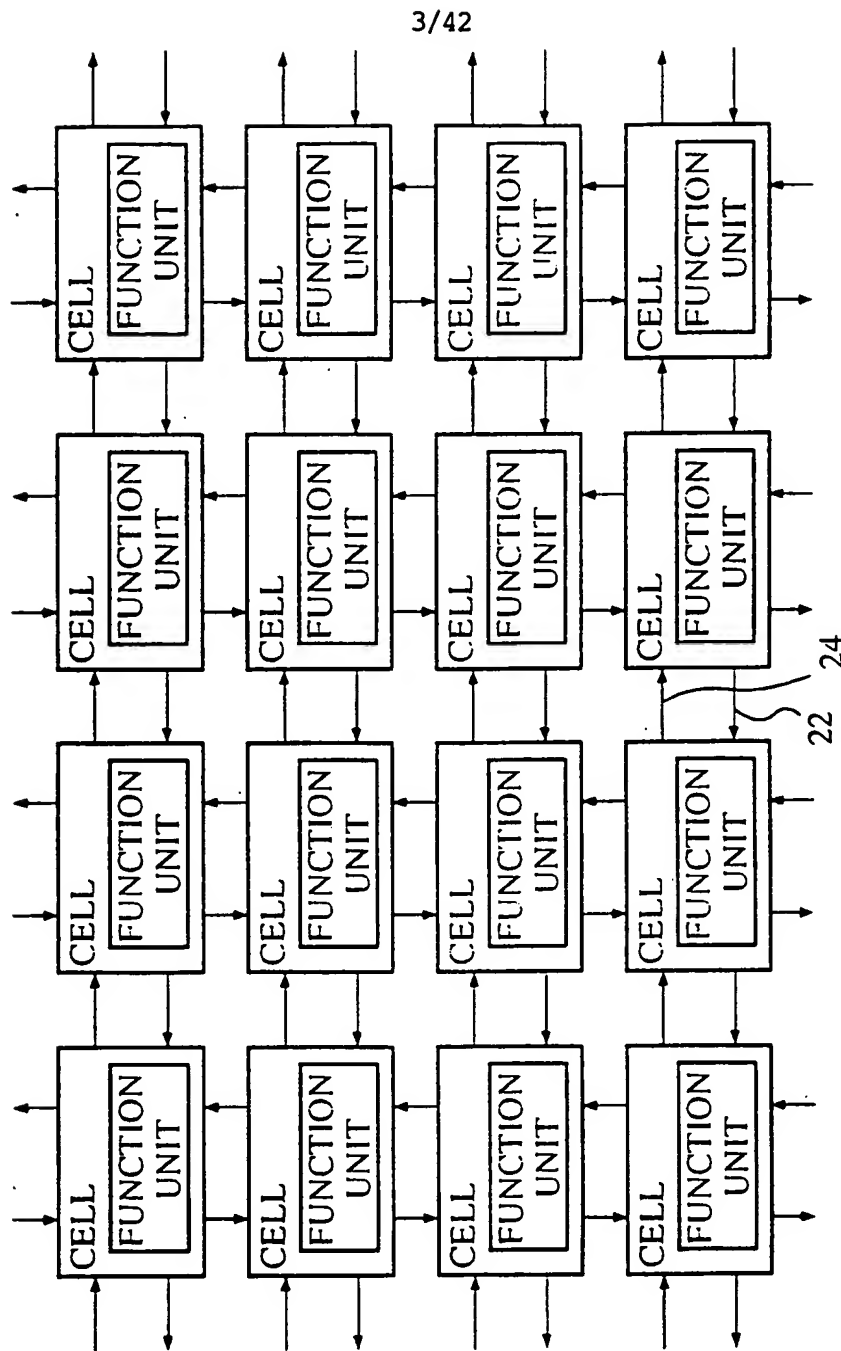


FIG. 3

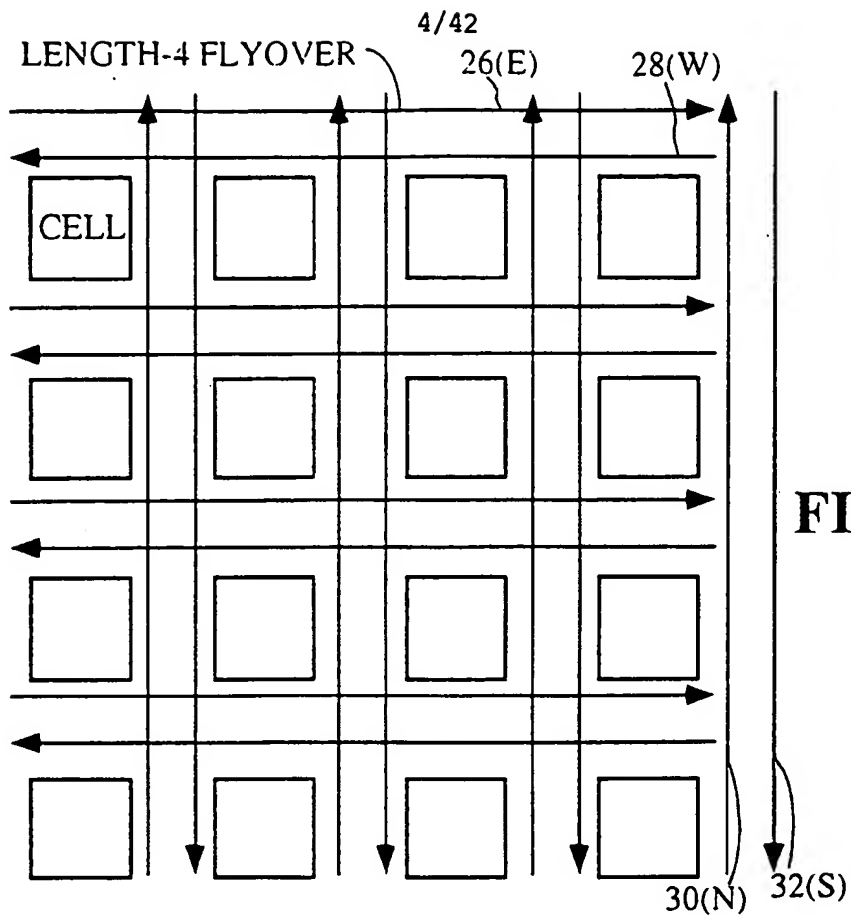


FIG. 4

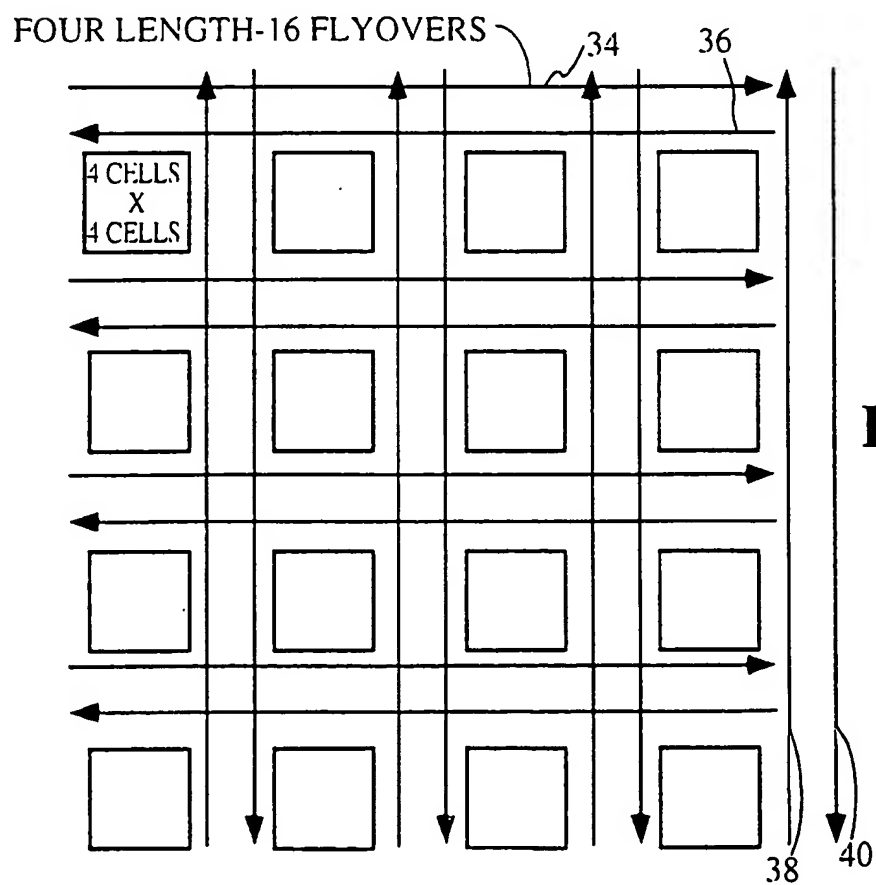


FIG. 5

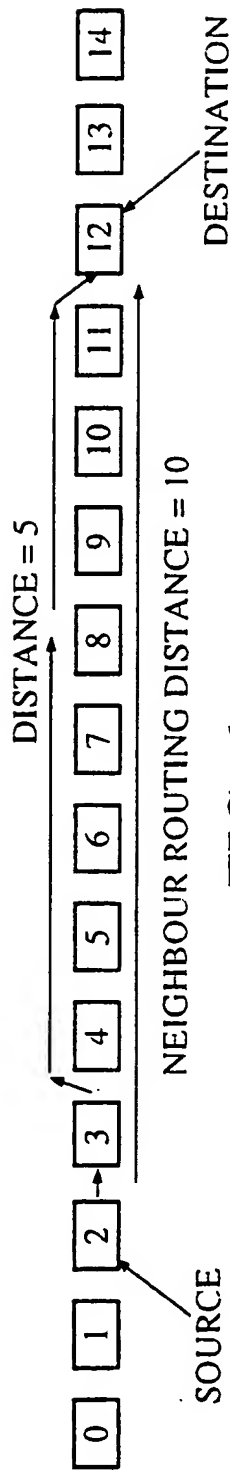


FIG. 6

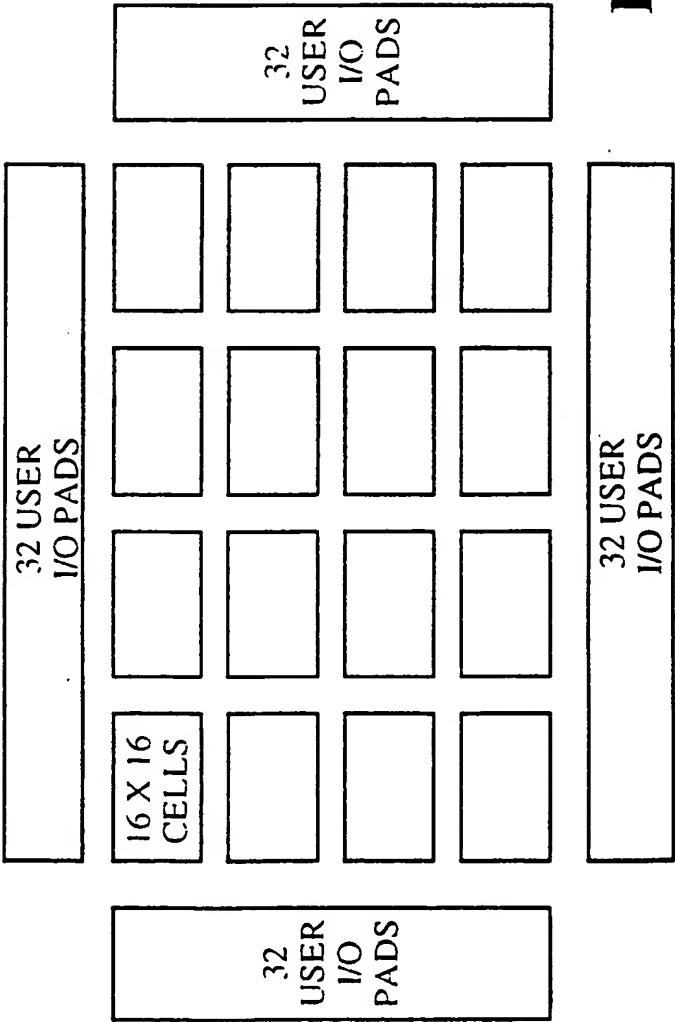
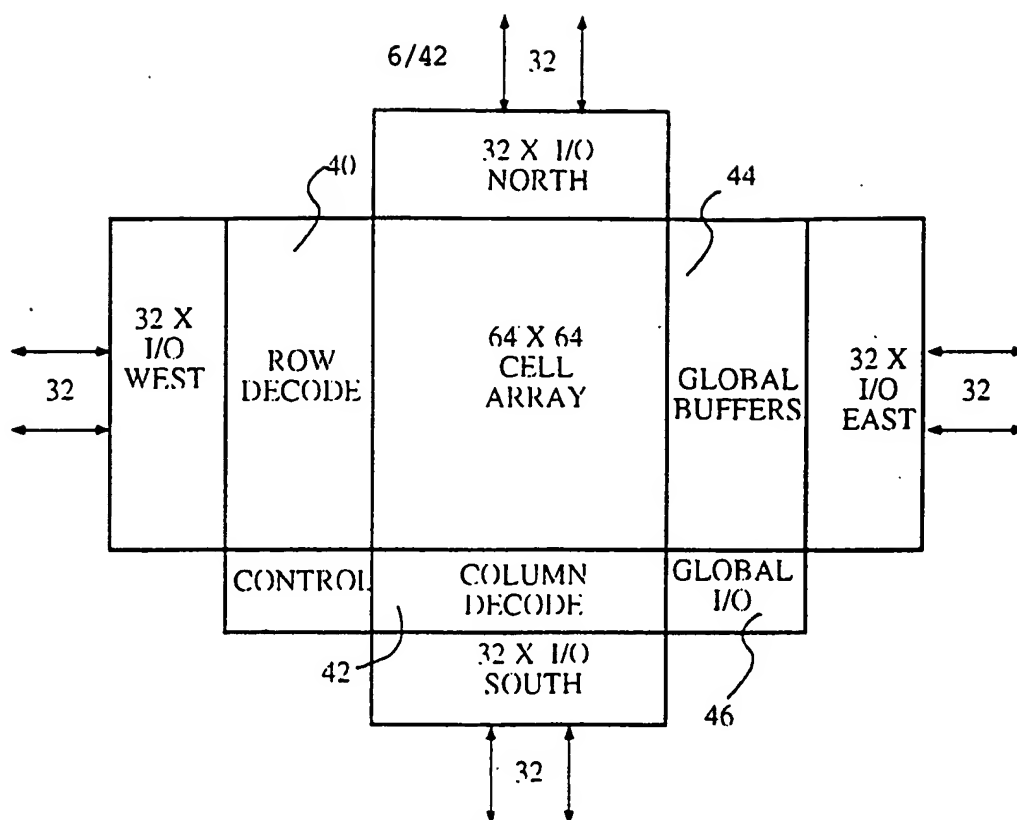
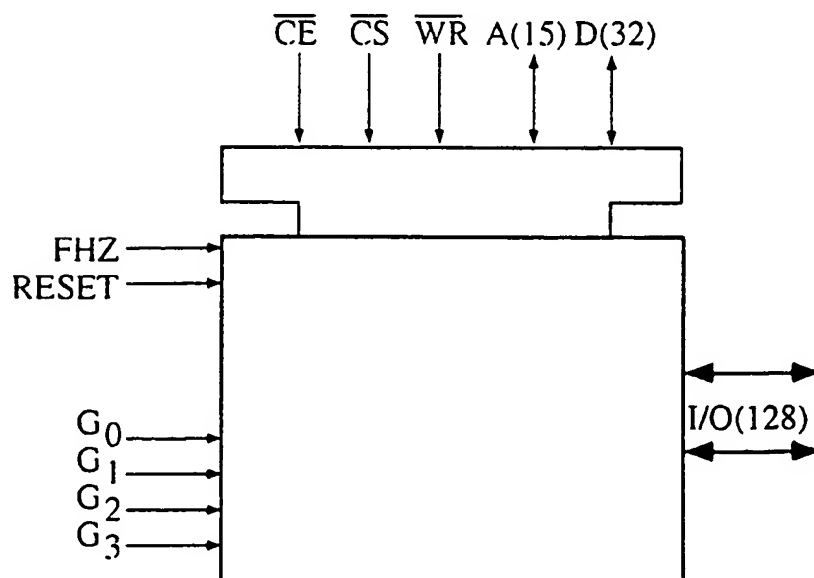


FIG. 7

**FIG. 8****FIG. 9**

7/42

FIG. 10

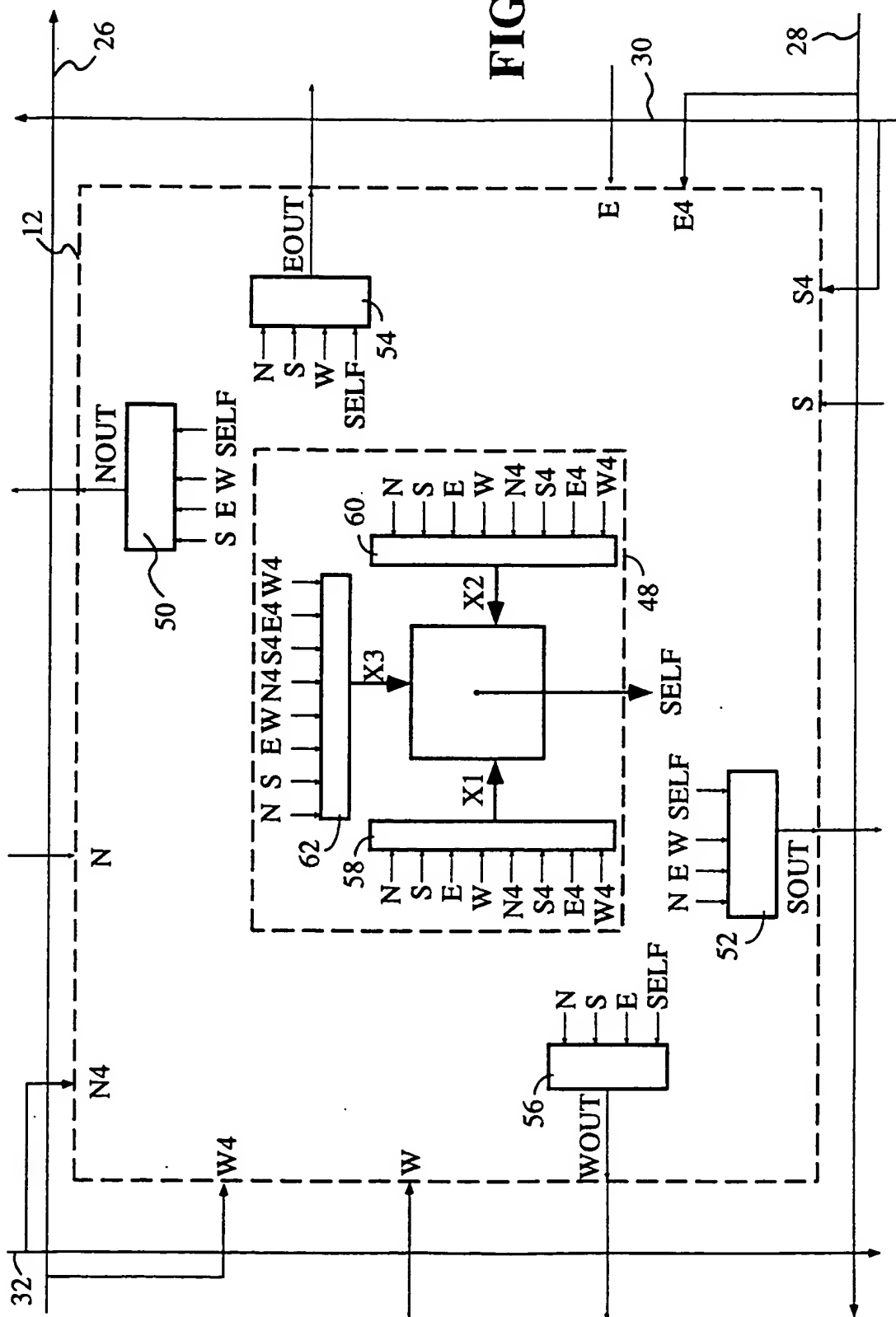
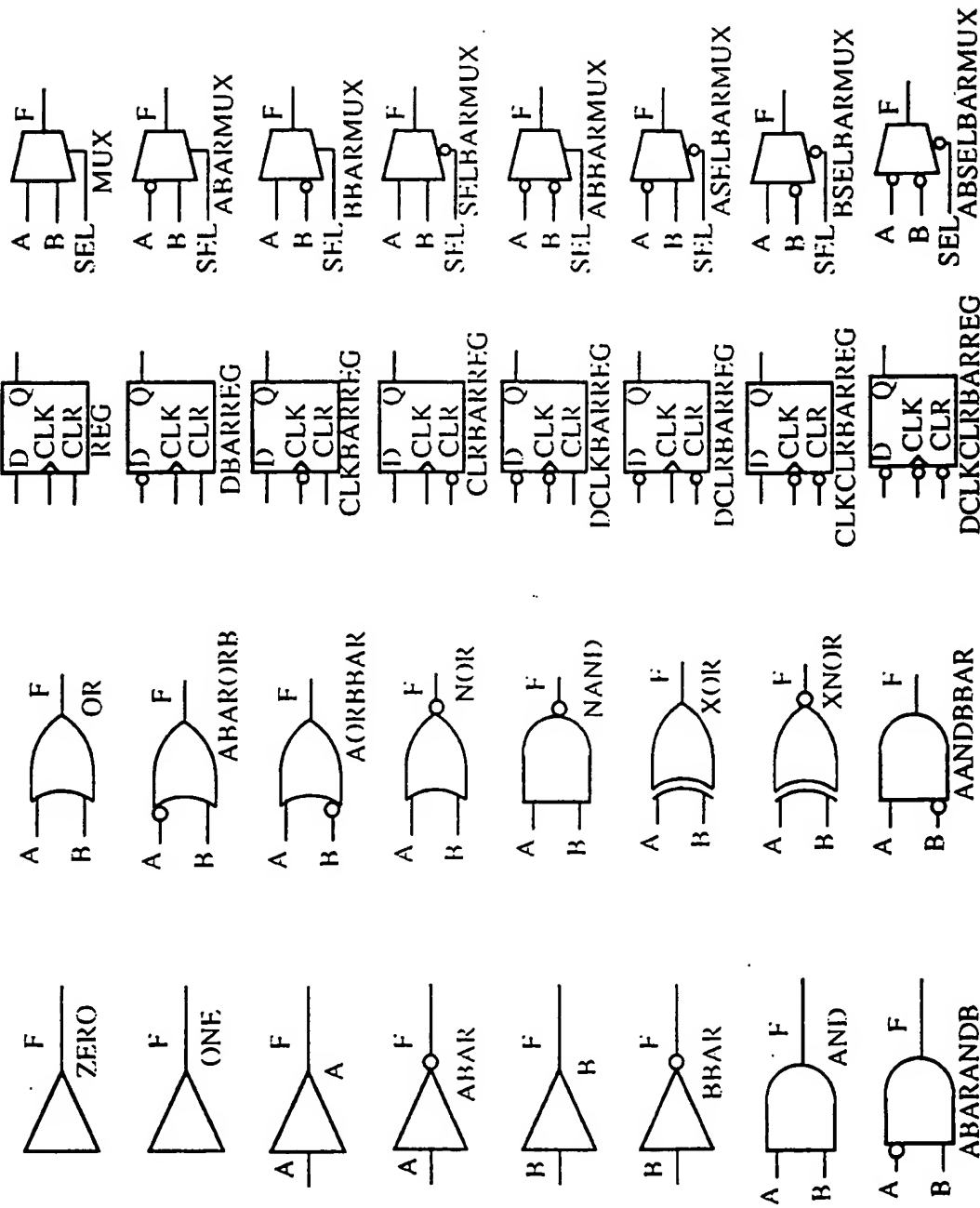


FIG. 11



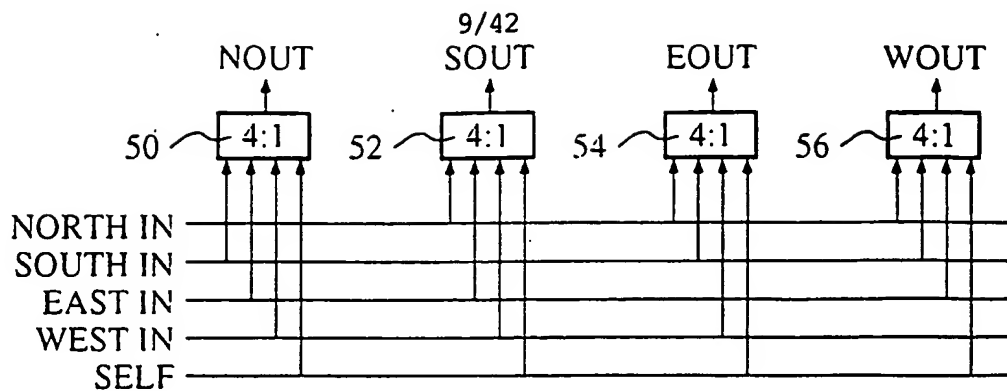


FIG. 12

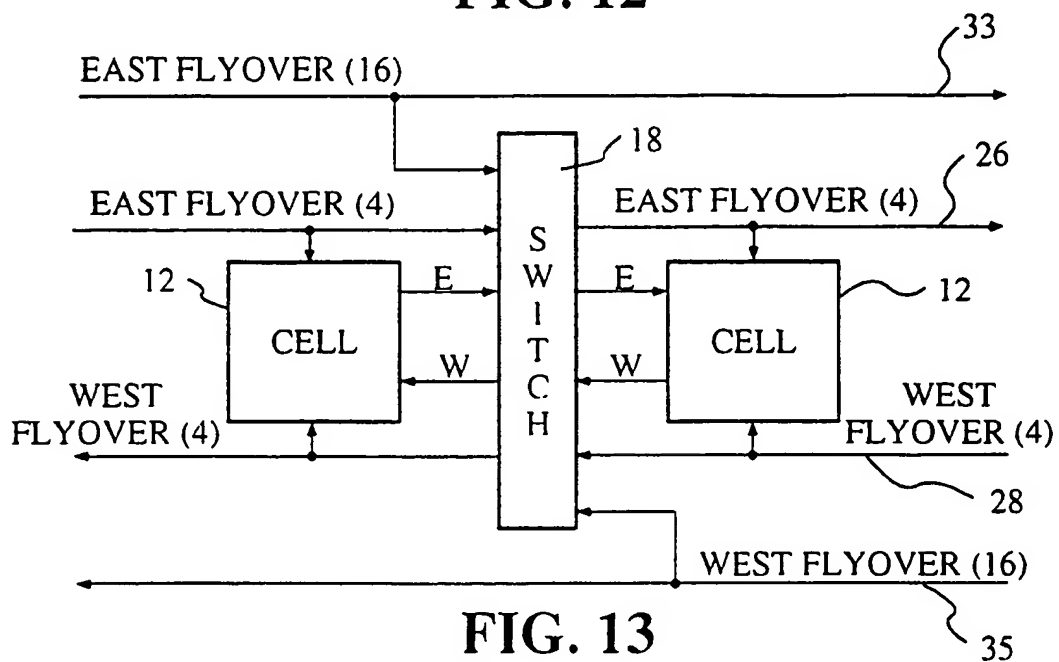


FIG. 13

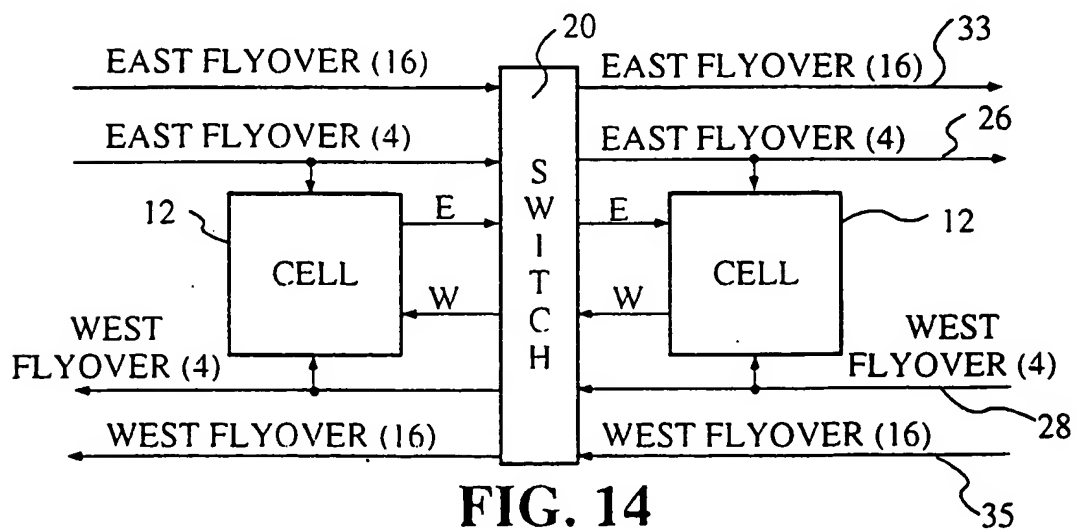


FIG. 14

10/42

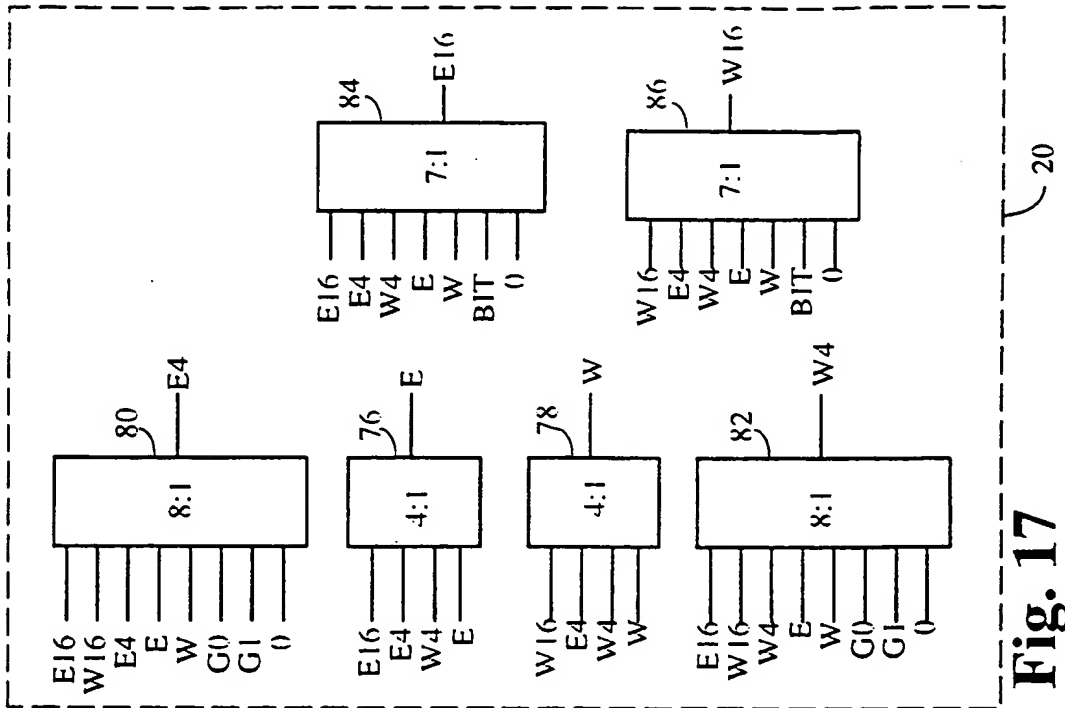


Fig. 17

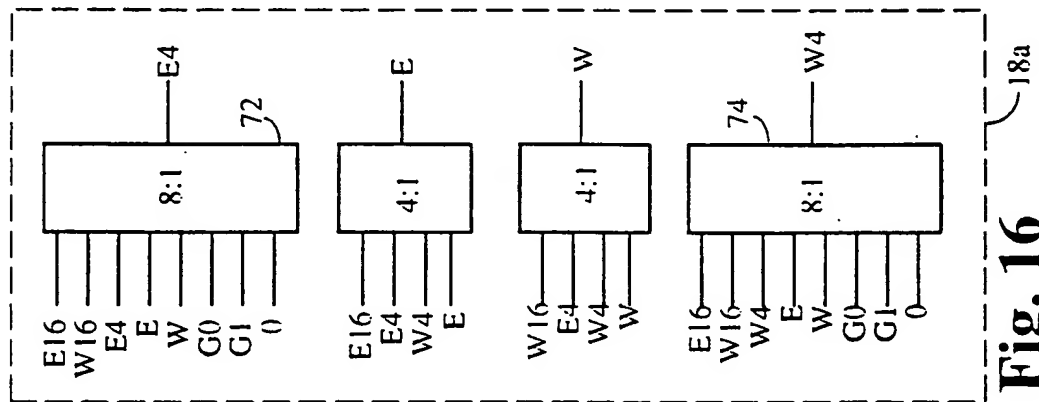


Fig. 16

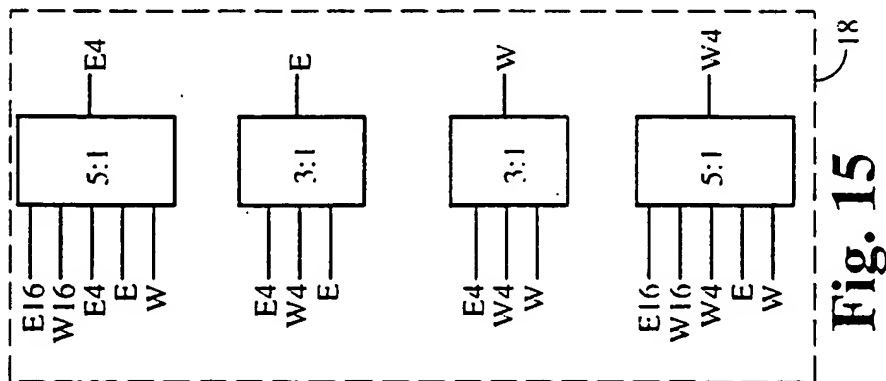


Fig. 15

11/42

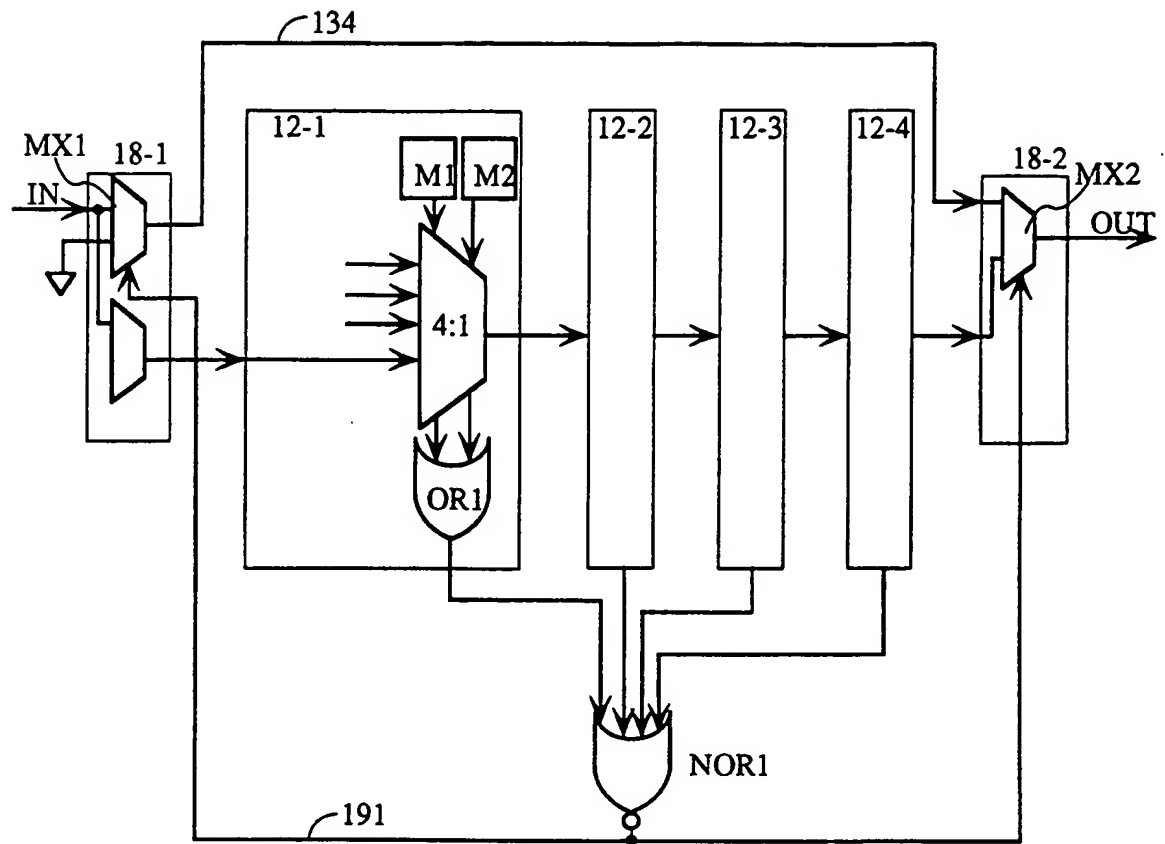


Fig. 18

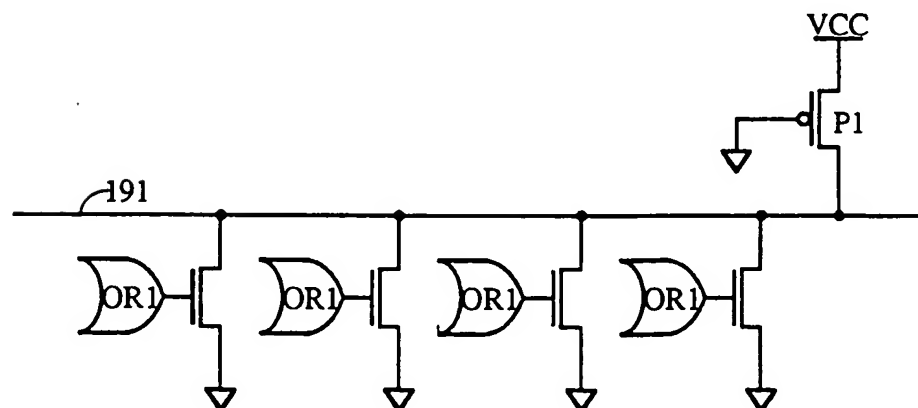


Fig. 19

12/42

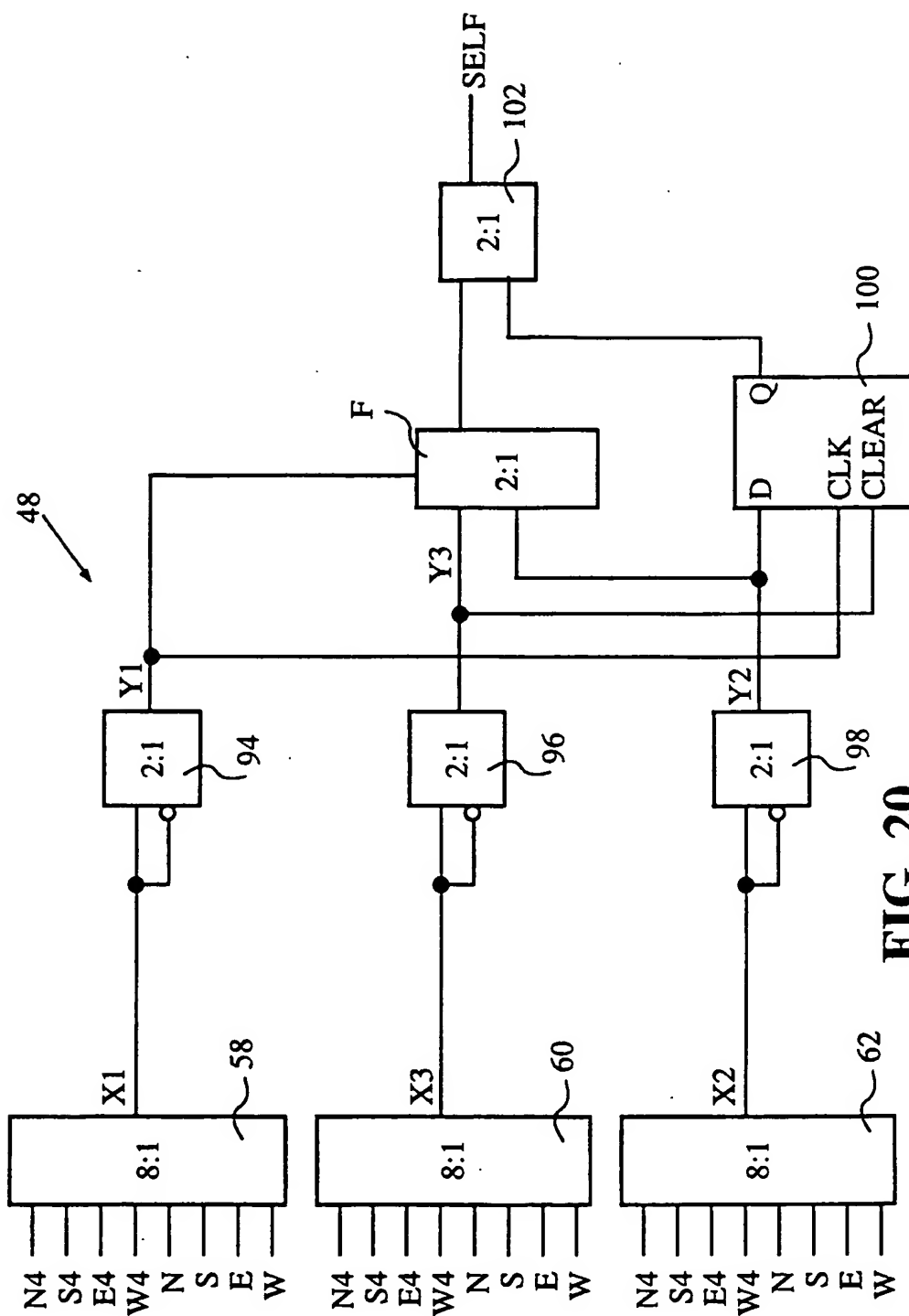


FIG. 20

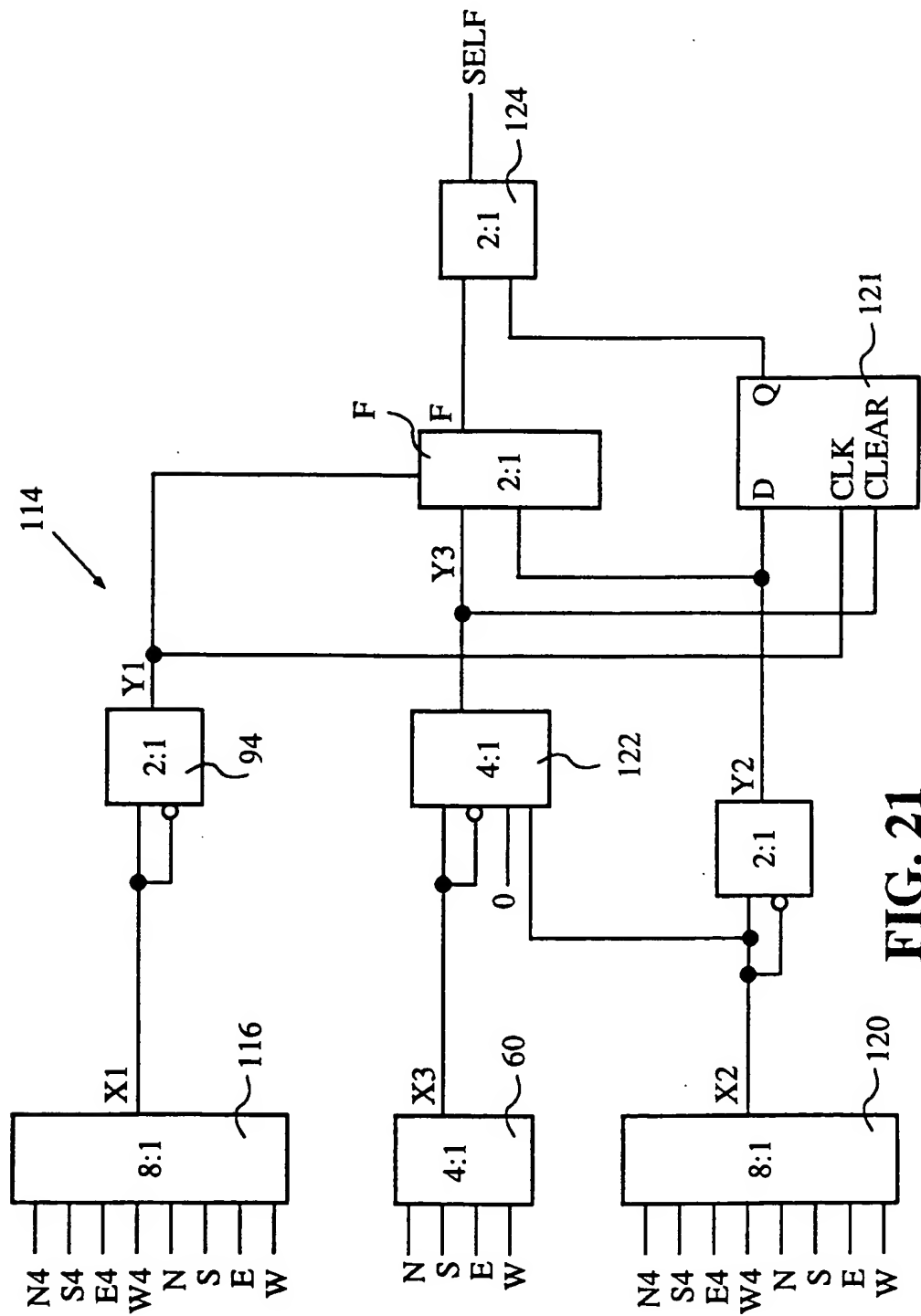


FIG. 21

14/42

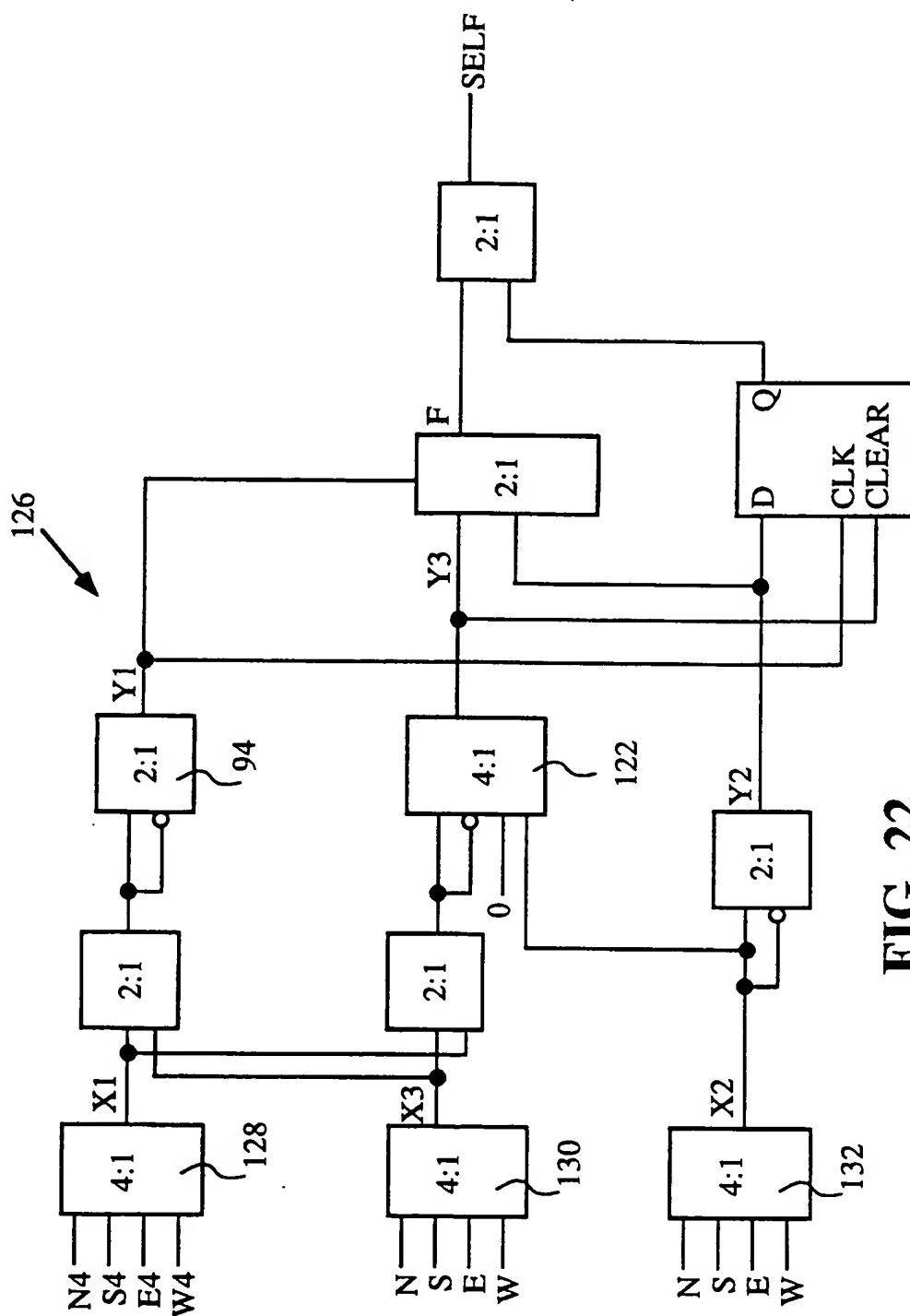


FIG. 22

15/42

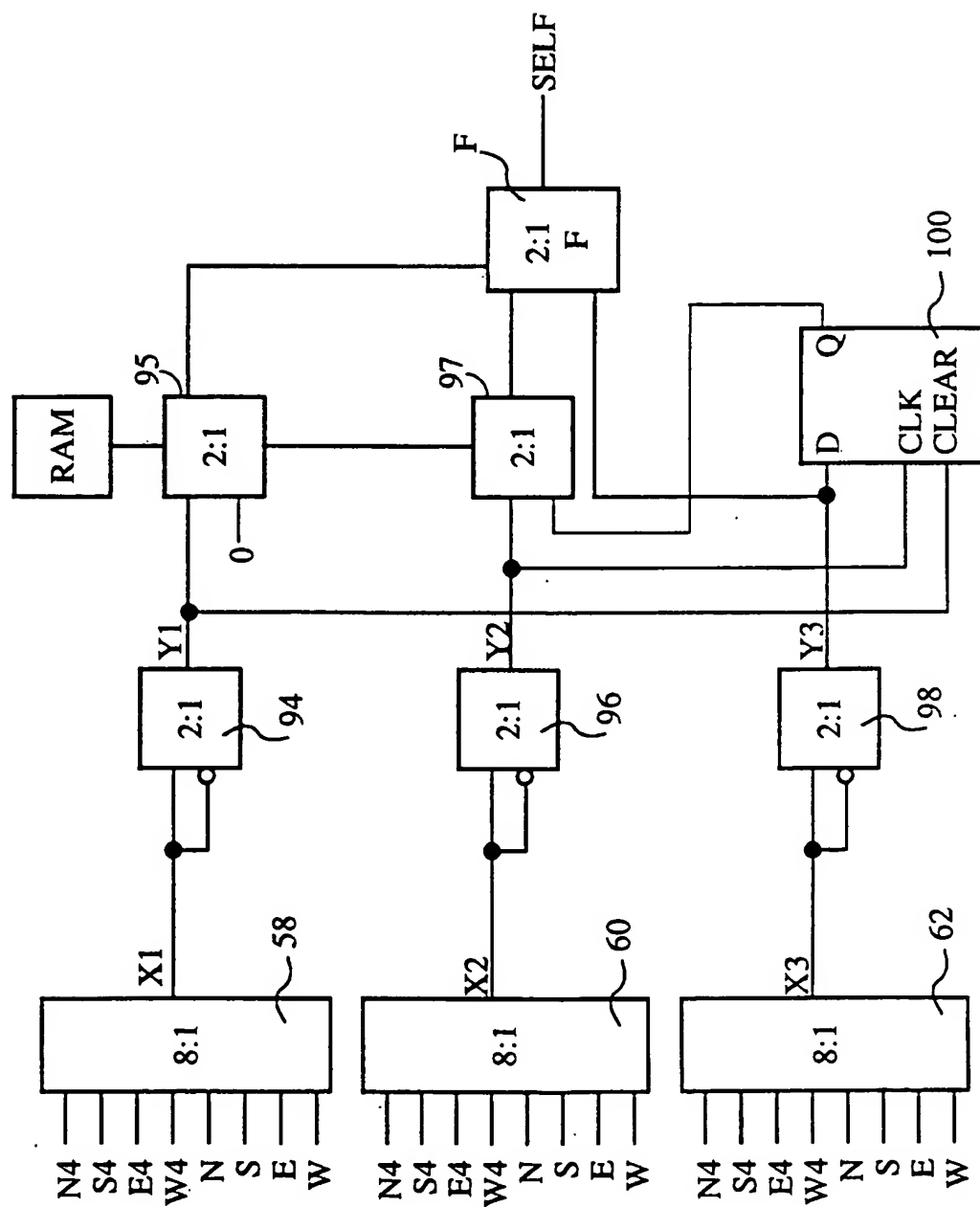


FIG. 23

16/42

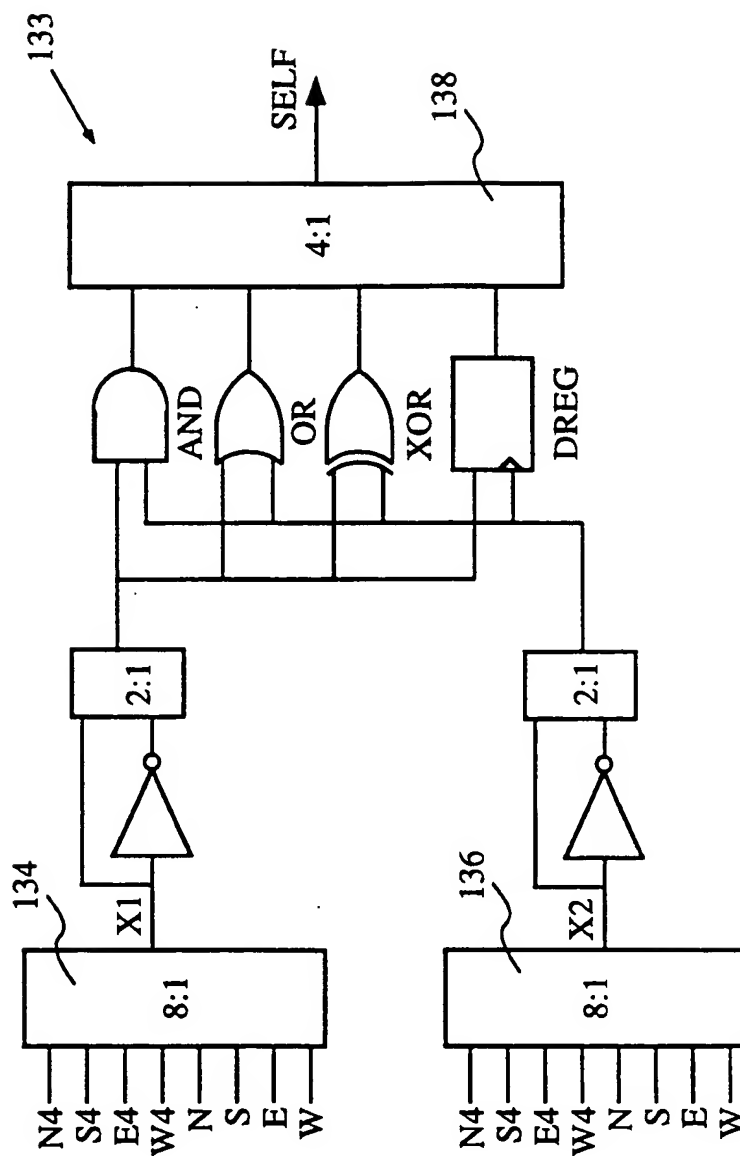


FIG. 24

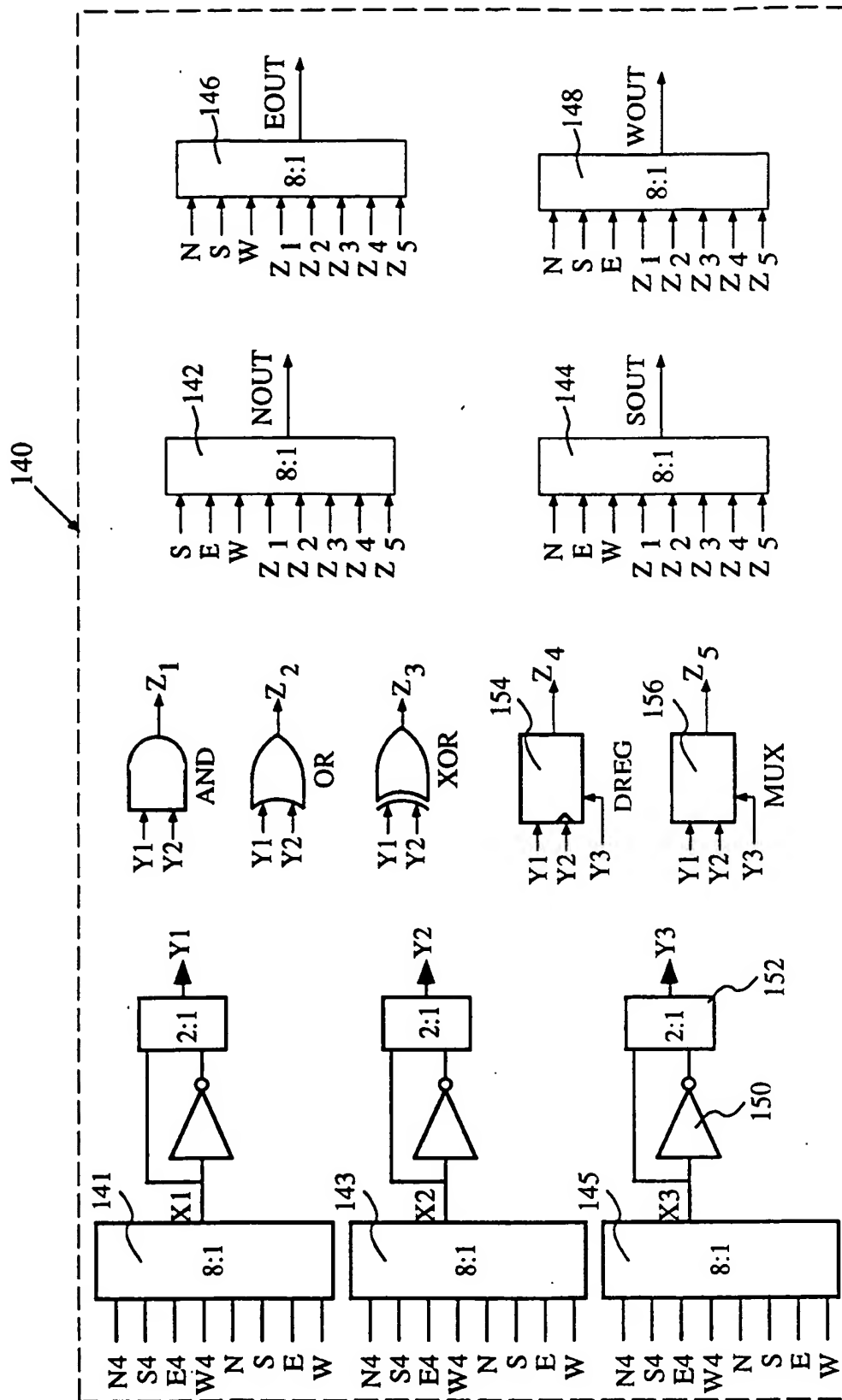


FIG. 25

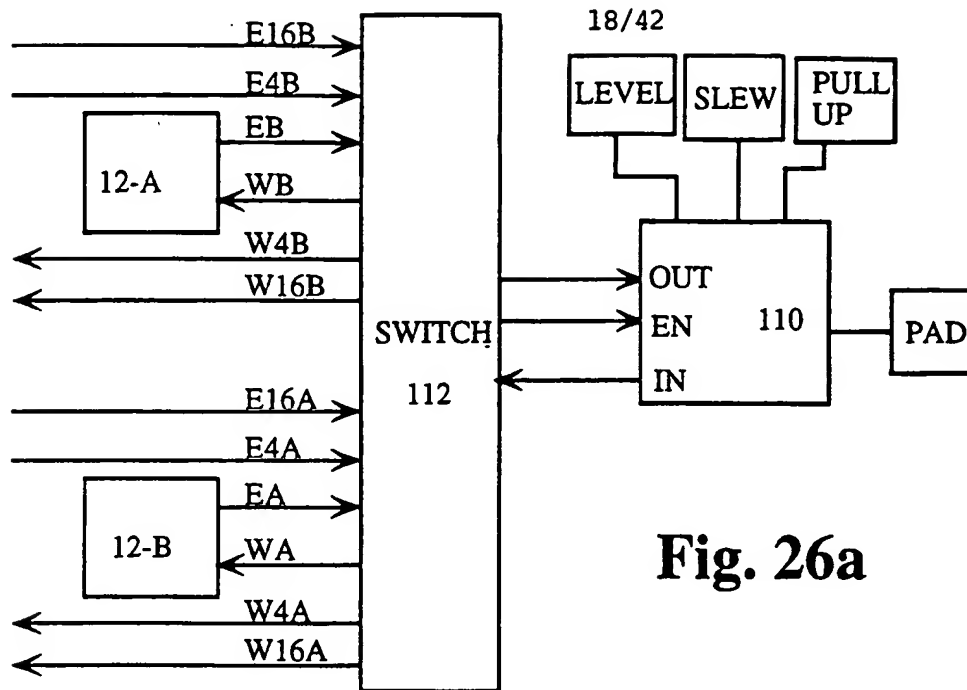


Fig. 26a

PARAMETER	VALUE
MODE	INPUT, OUTPUT, TRISTATE, OPEN DRAIN
INPUT LEVEL	TTL, CMOS
SLEW RATE	FAST, SLOW

Fig. 26b

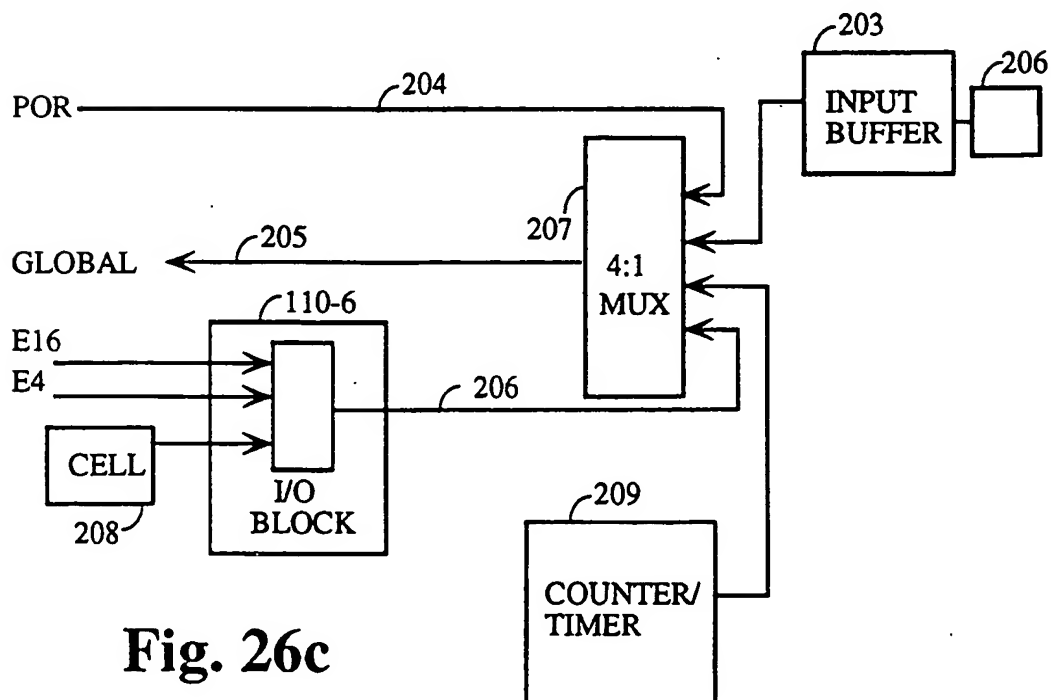


Fig. 26c

19/42

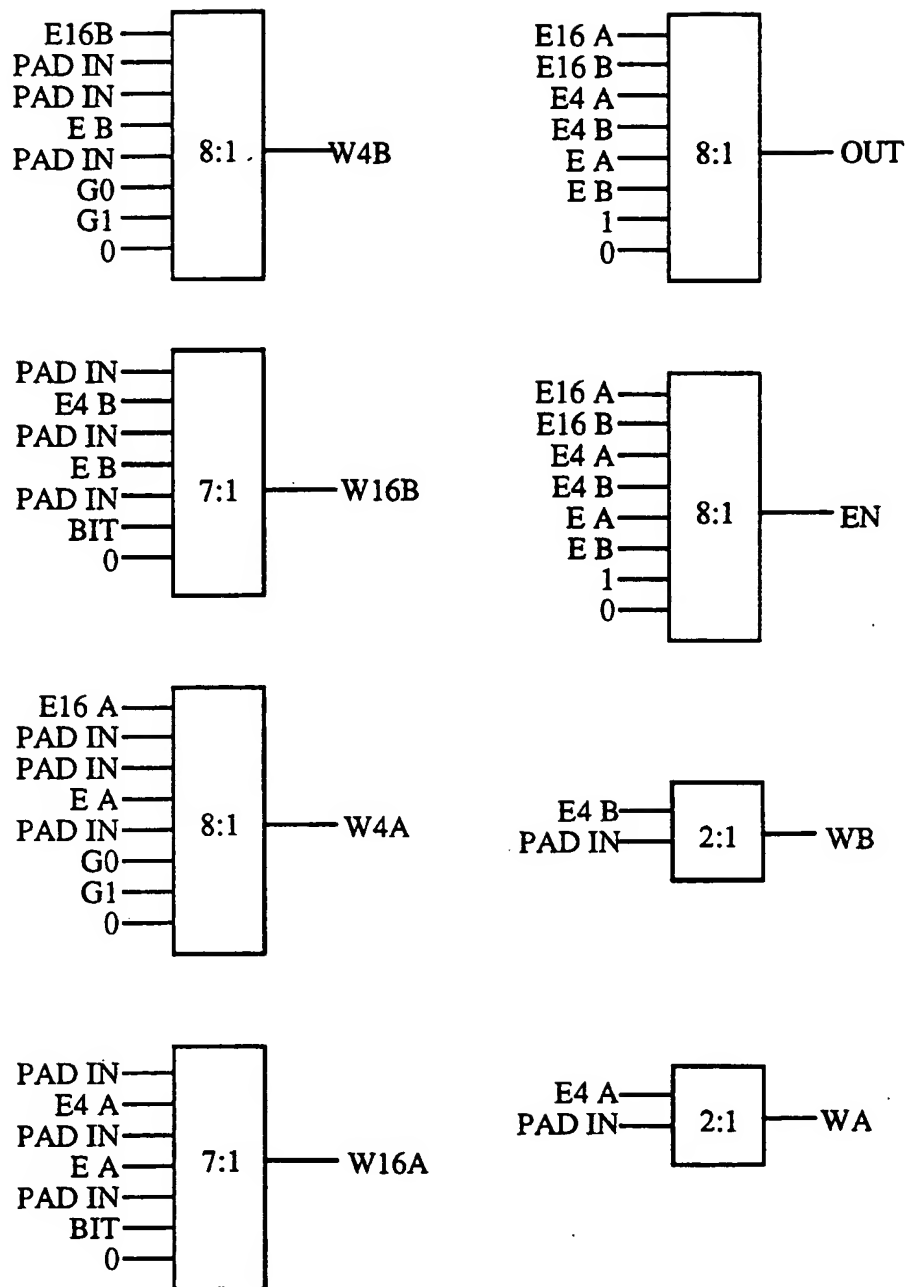


FIG. 26d

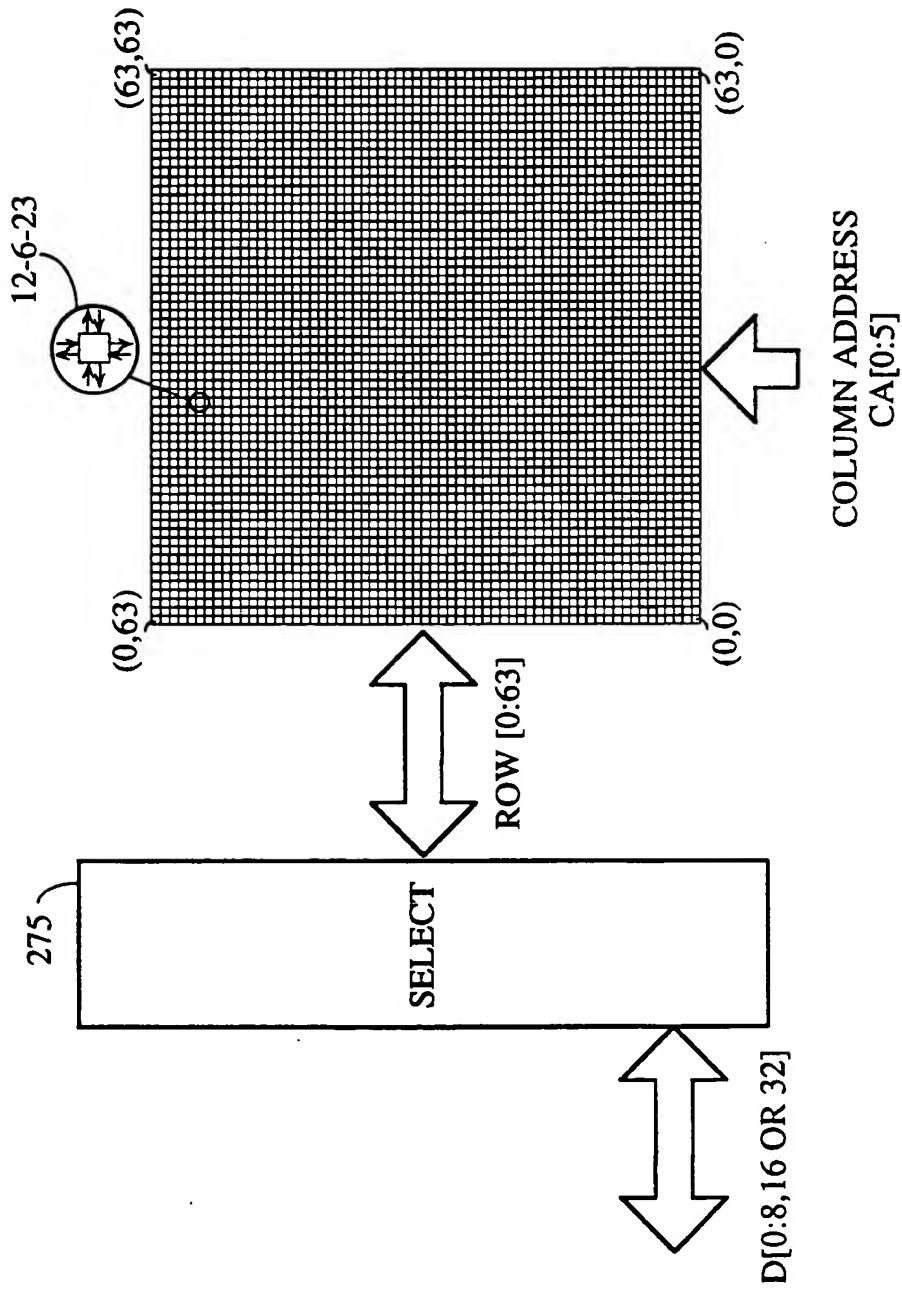
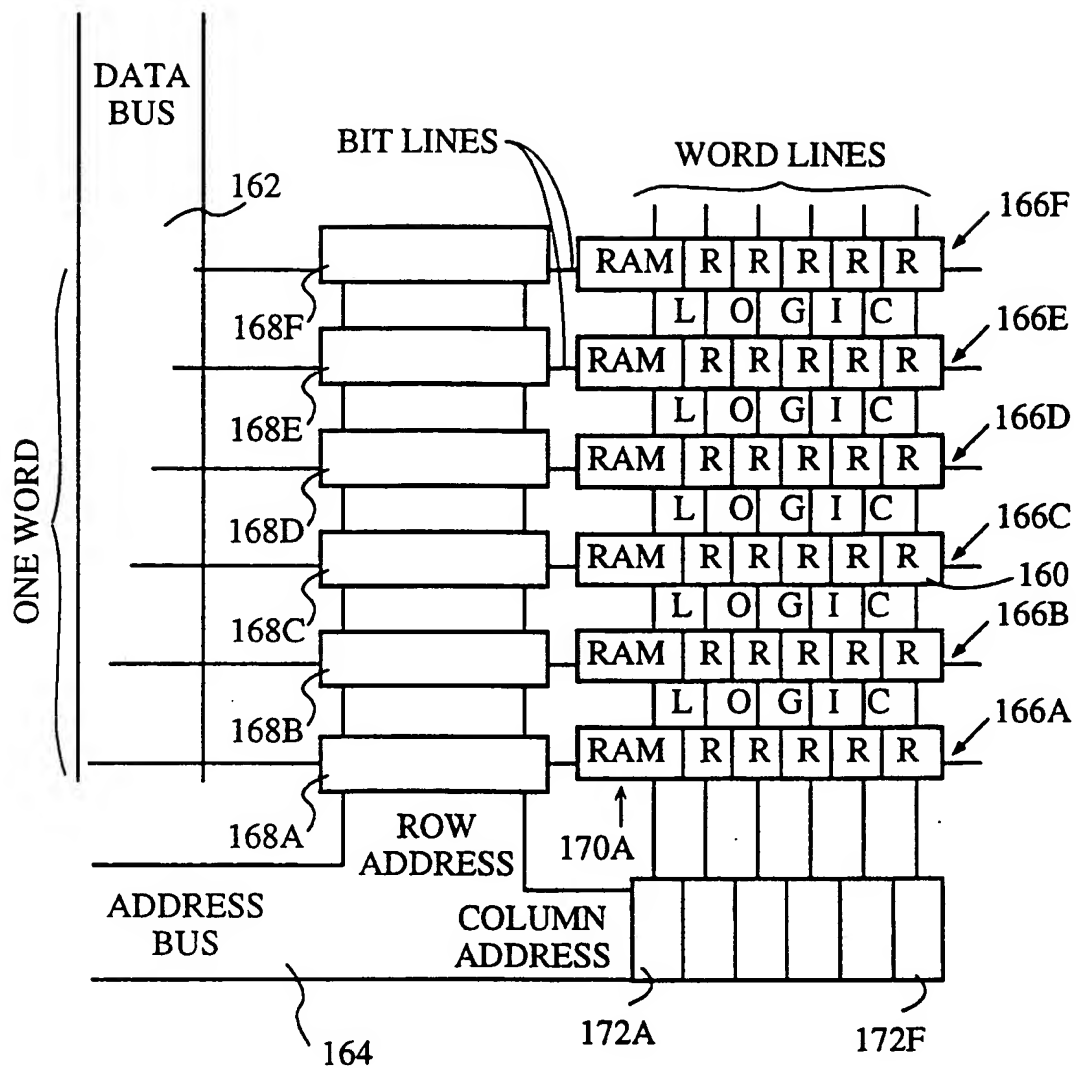


FIG. 27a

21/42

**FIG. 27b**

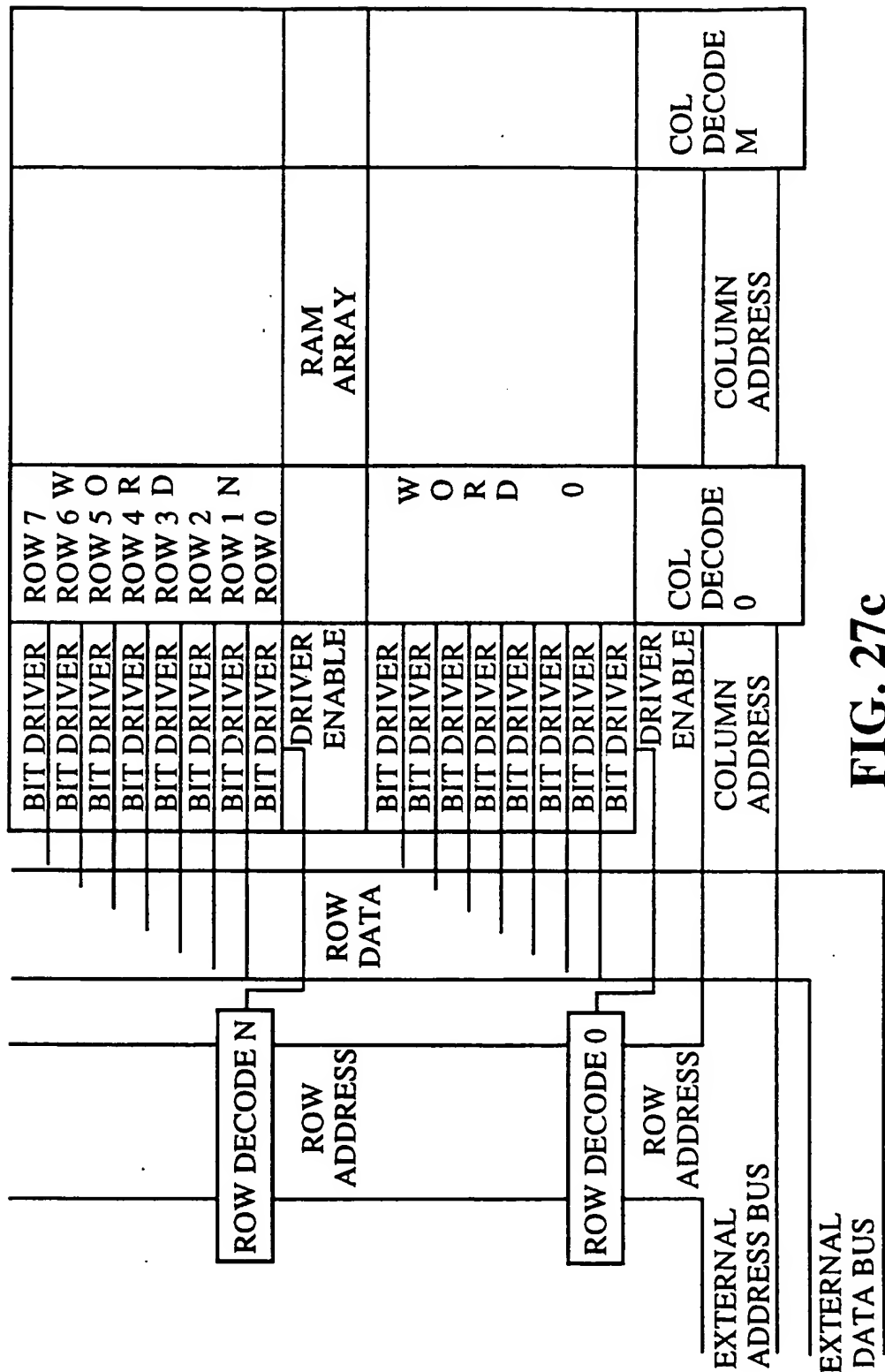


FIG. 27c

23/42

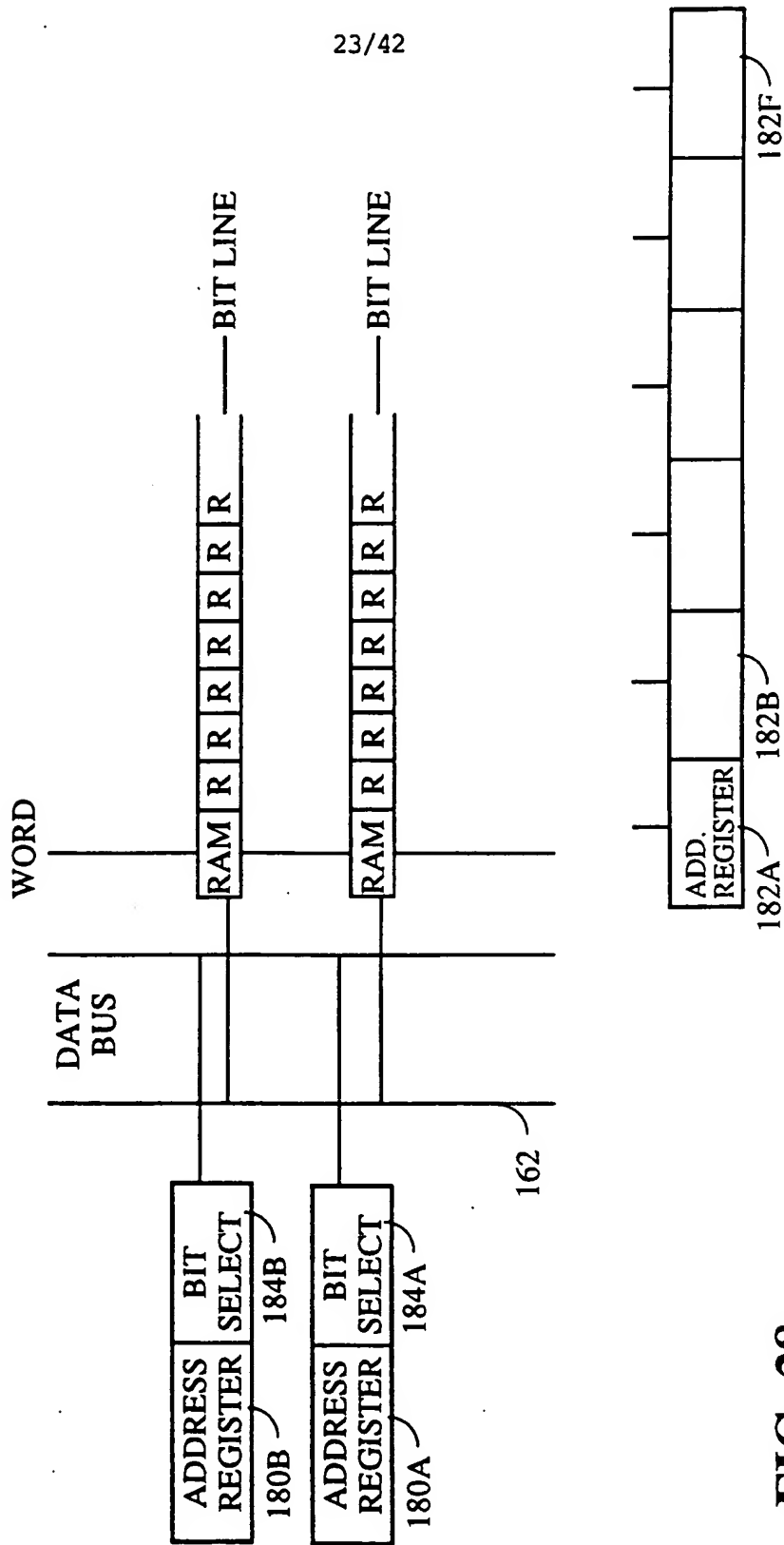


FIG. 28

24/42

CELL COLUMN	CELL ROW	SIDE	MODE
0:5	6:11	12	13:14

FIG. 29

MODE0	MODE1	STORE SELECTED
0	0	CELL ROUTING
0	1	CELL FUNCTION
1	0	CHANNELS, I/O
1	1	STATE ACCESS, DEVICE CONFIG.

FIG. 30

CELL ROUTING MODE

BIT	0	1	2	3	4	5	6	7
SIDE 0	SOUTH	WEST	EAST	NORTH				

FIG. 31a

CELL FUNCTION MODE

BIT	0	1	2	3	4	5	6	7
SIDE 0	X1[0:1]		X2[0:1]		X3[0:1]		UNUSED	
SIDE 1	X1[2]	Y1	X2[2]	Y2	X3[2]	Y3	COMB/SEQ	UNUSED

FIG. 31b

BIT	0	1	2	3	4	5	6	7
CELL ROW	ROW + 2	ROW + 4	ROW + 6	ROW + 8	ROW + 10	ROW + 12	ROW + 14	

FIG. 35

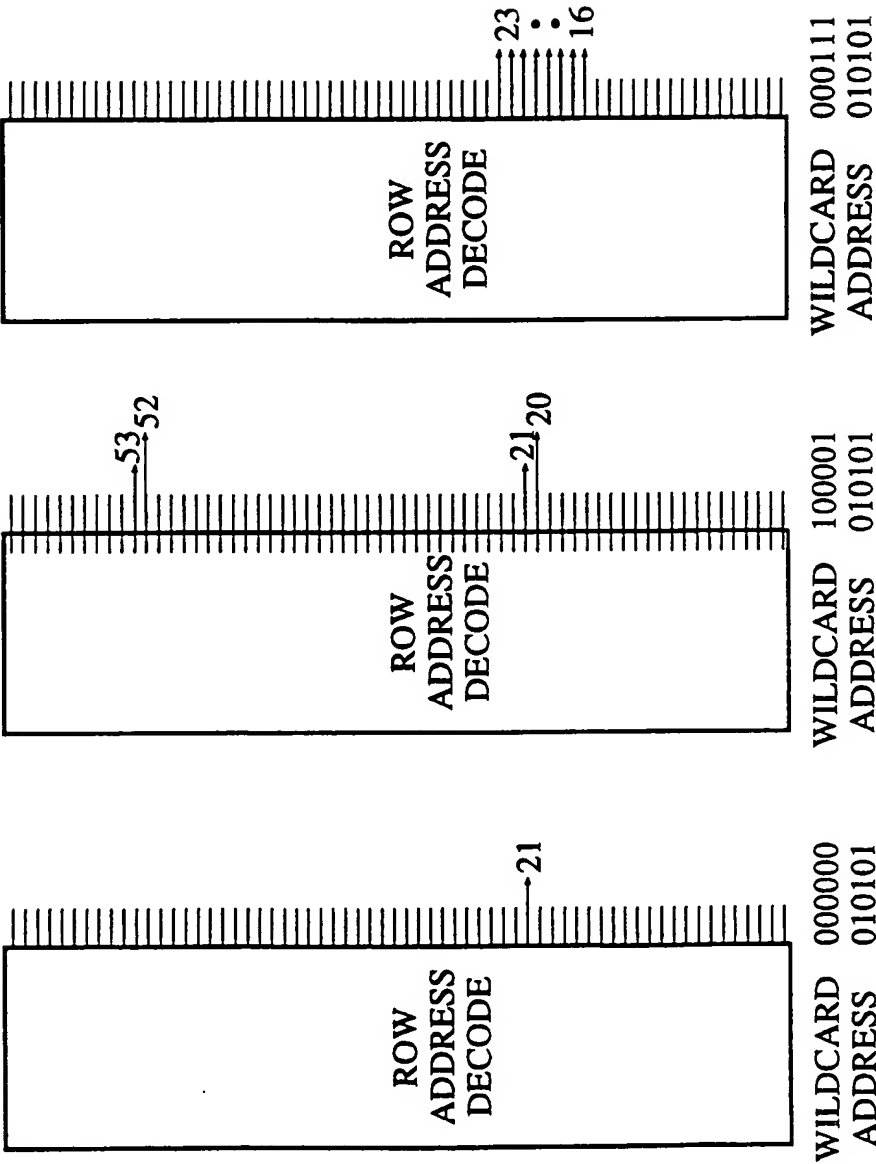
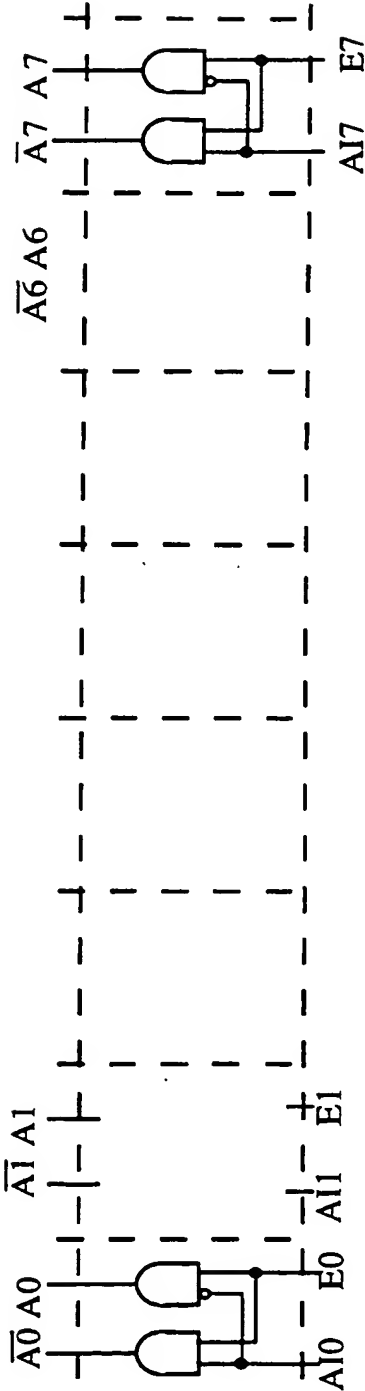
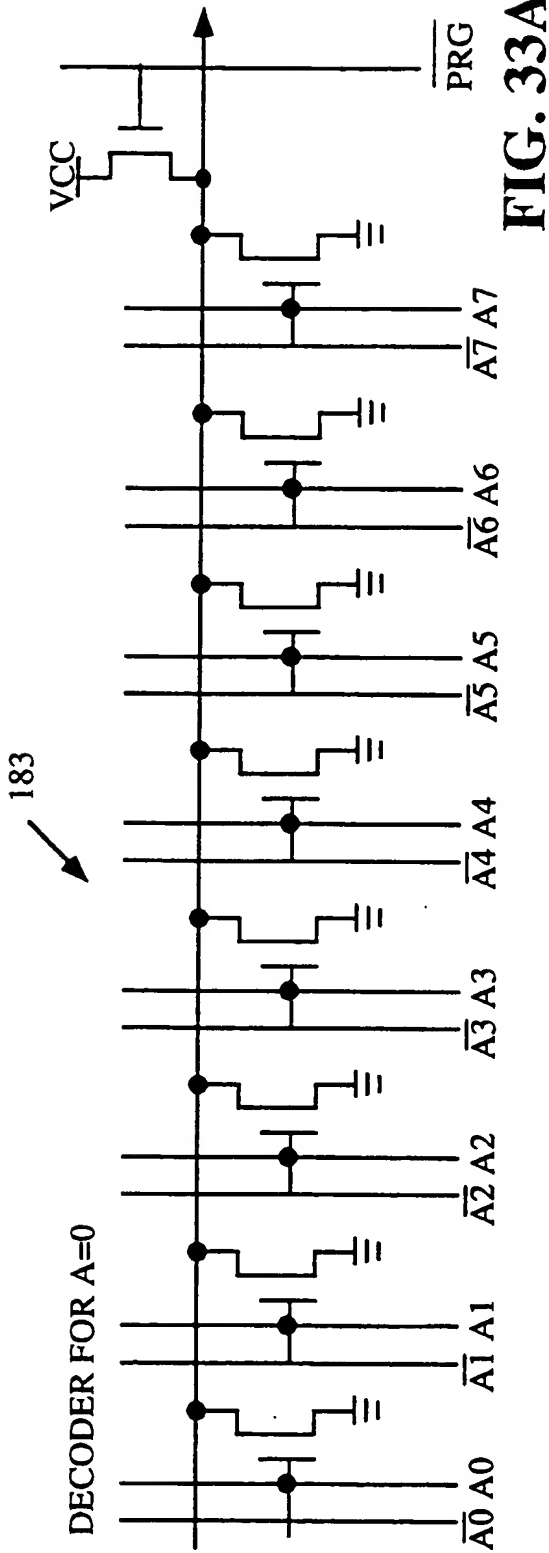


Fig. 32c

Fig. 32b

Fig. 32a

FIG. 32



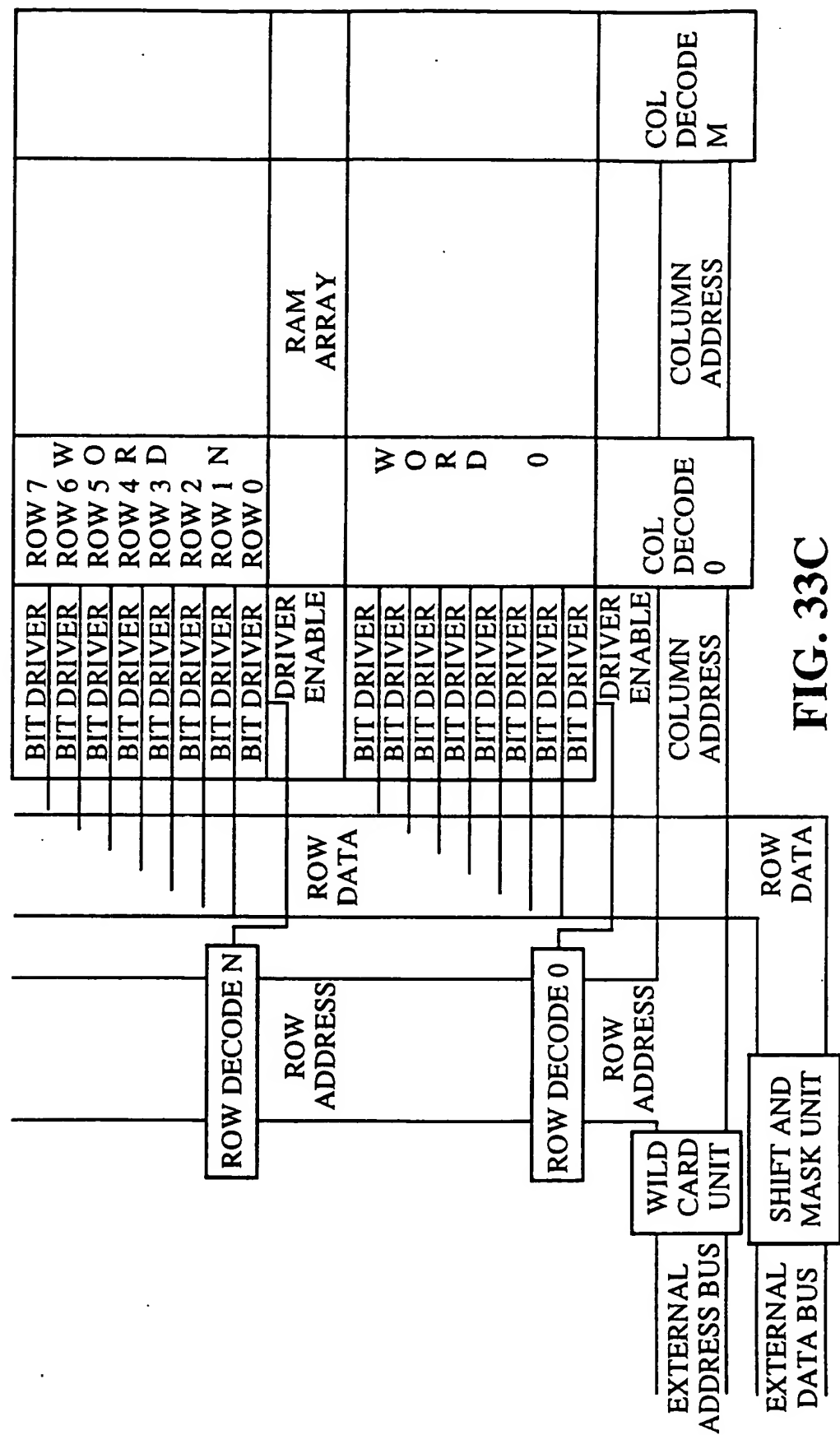


FIG. 33C

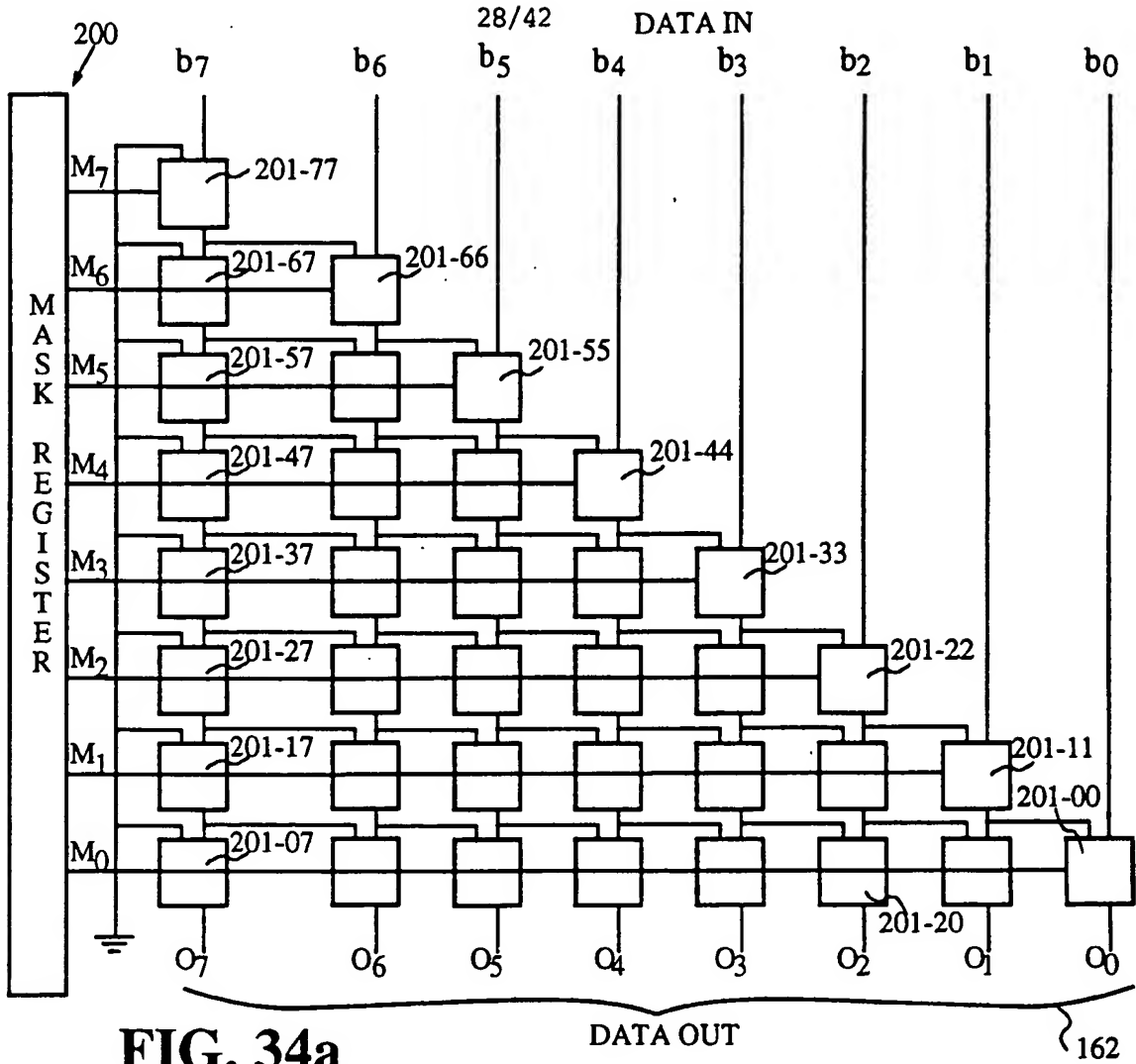


FIG. 34a

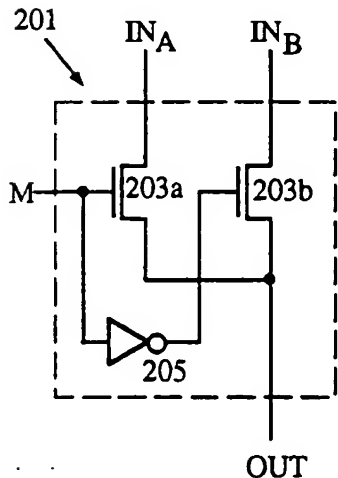


FIG. 34b

Internal Bit	b7	b6	b5	b4	b3	b2	b1	b0	DATA IN
Mask Register	1	0	0	1	0	0	0	1	ENABLE
External Bit	0	0	0	b6	b5	b3	b2	b1	DATA OUT

FIG. 34c

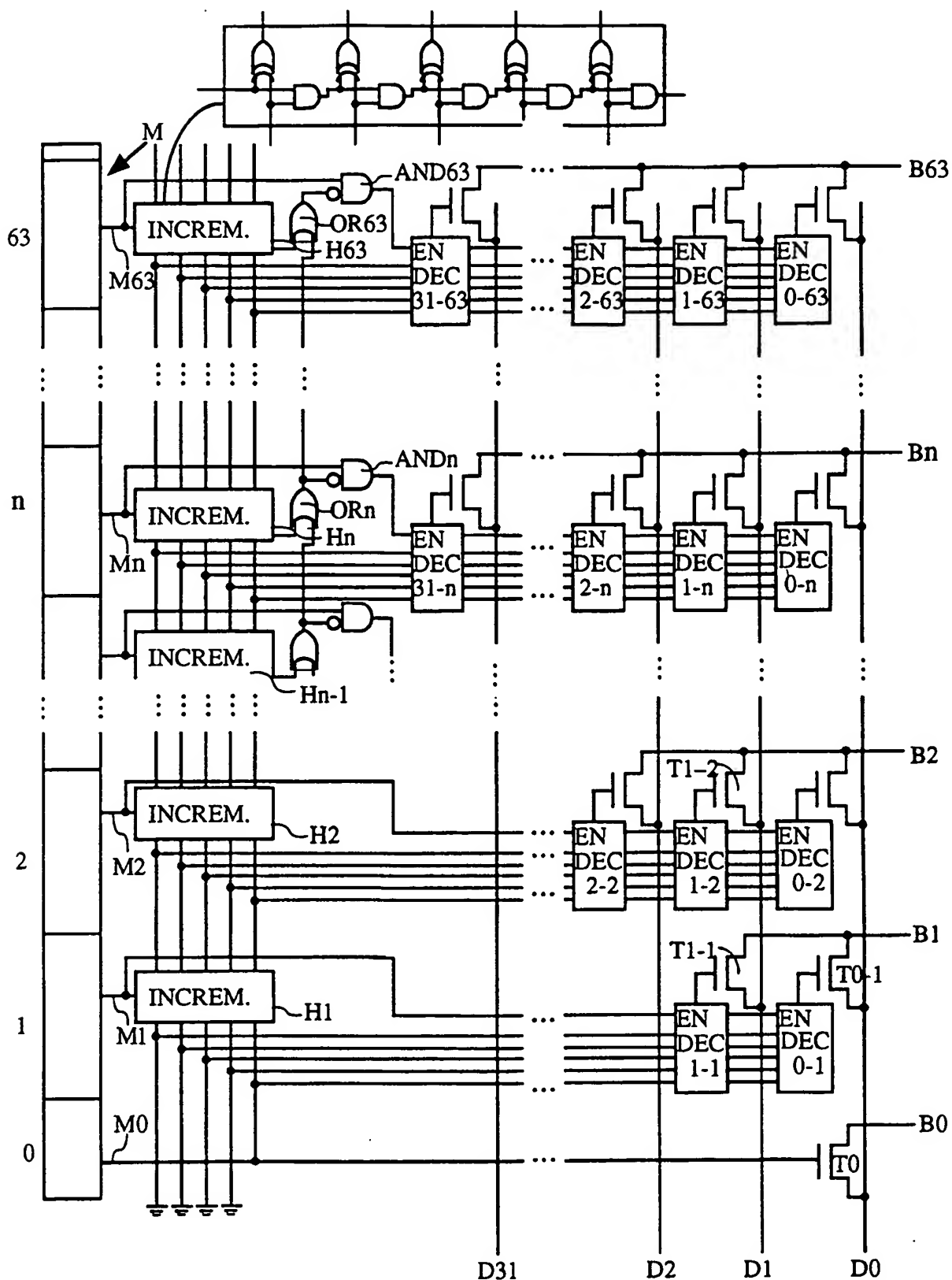


FIG. 36a

30/42

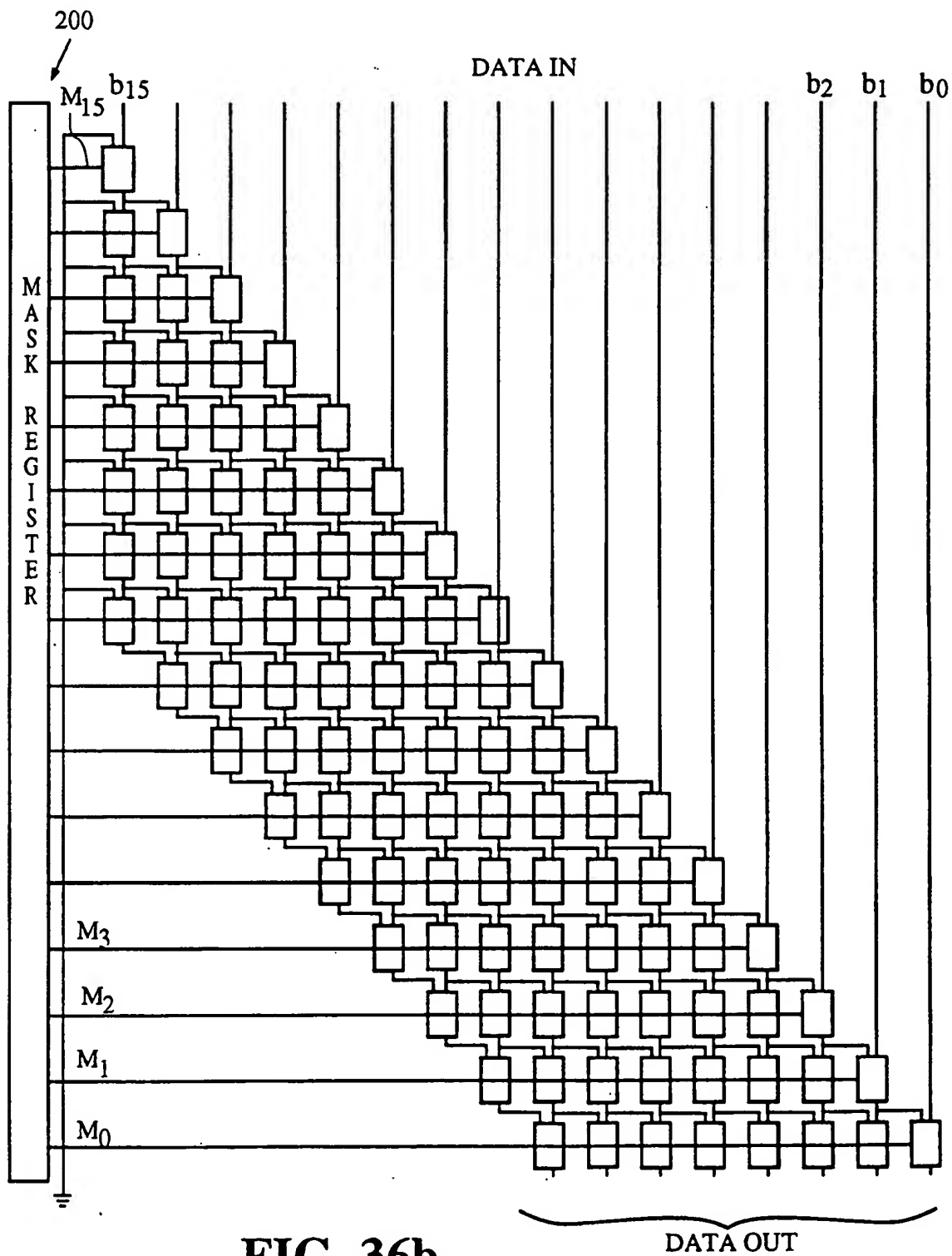


FIG. 36b

31/42

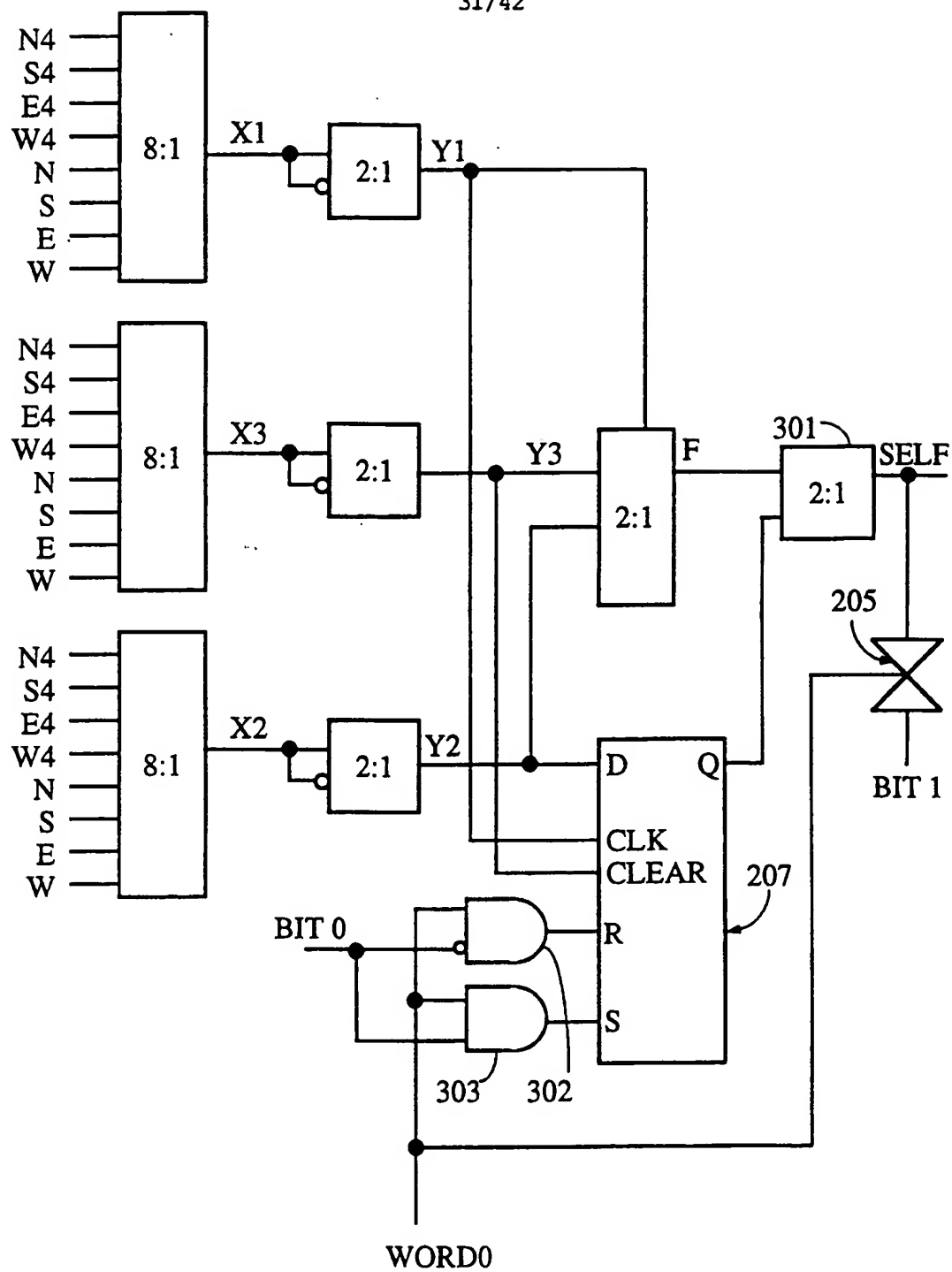
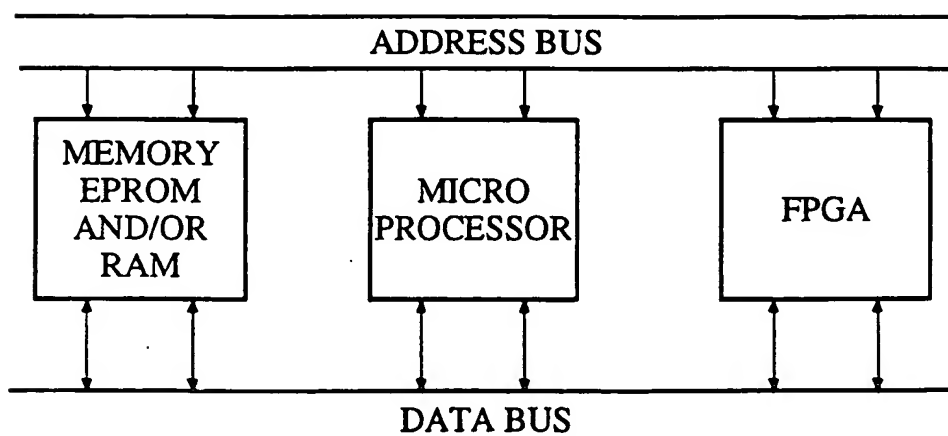
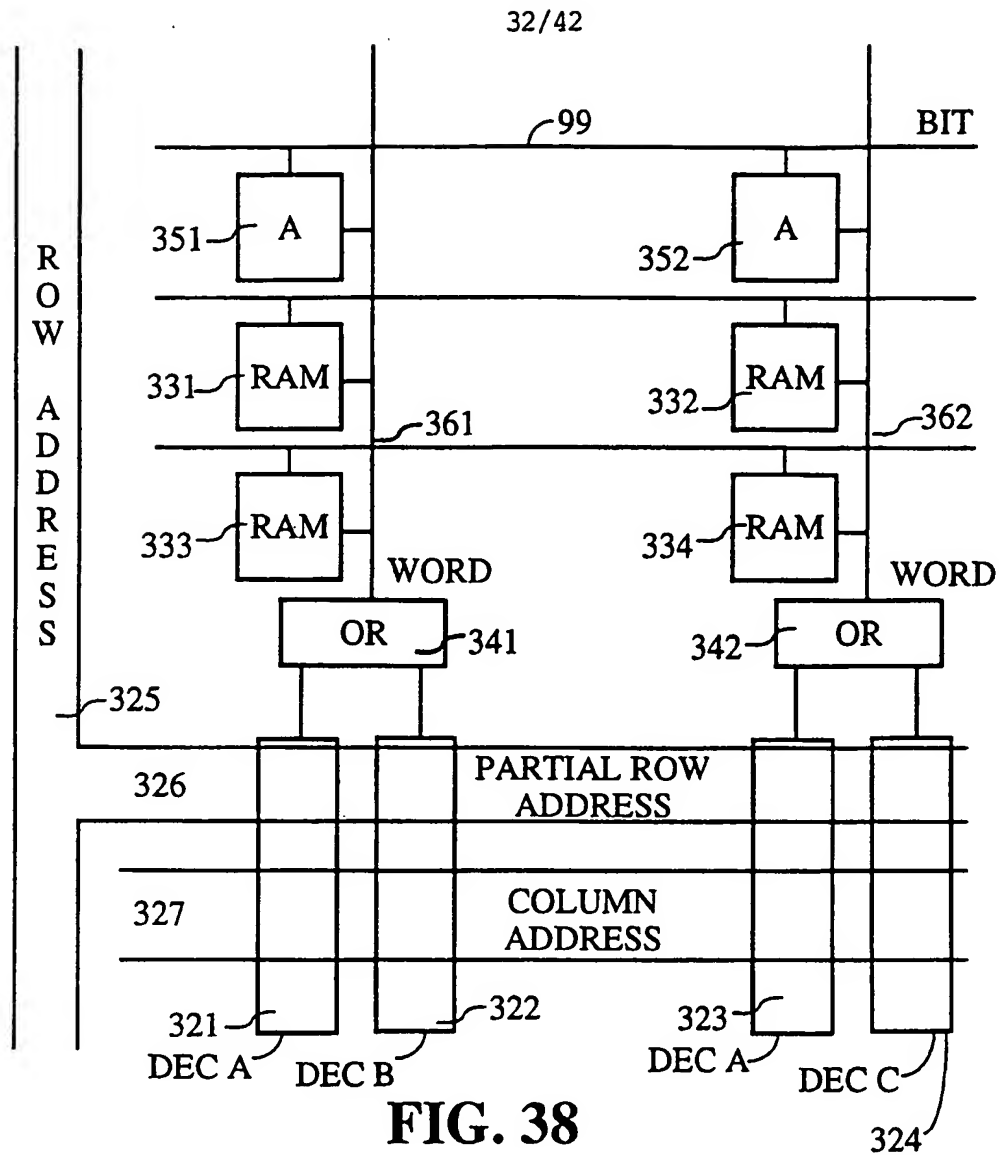


FIG. 37



33/42

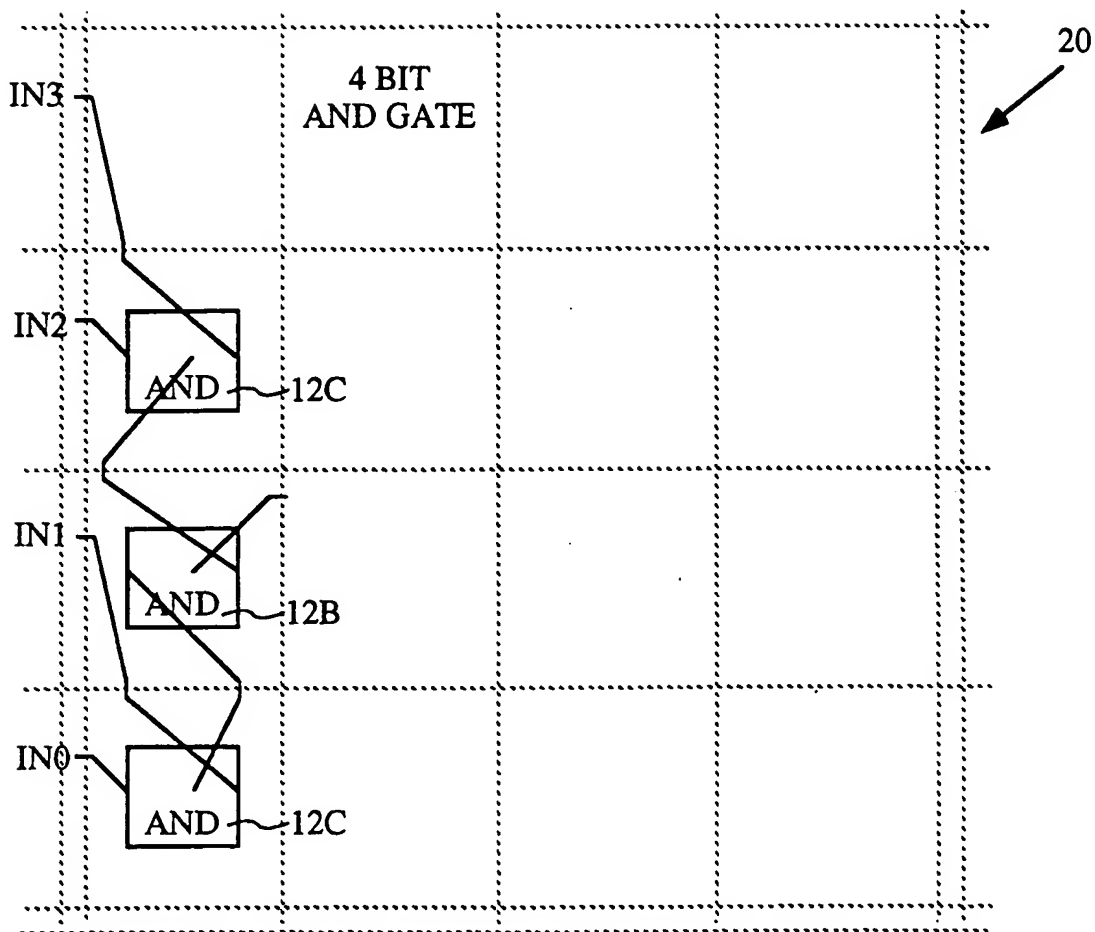
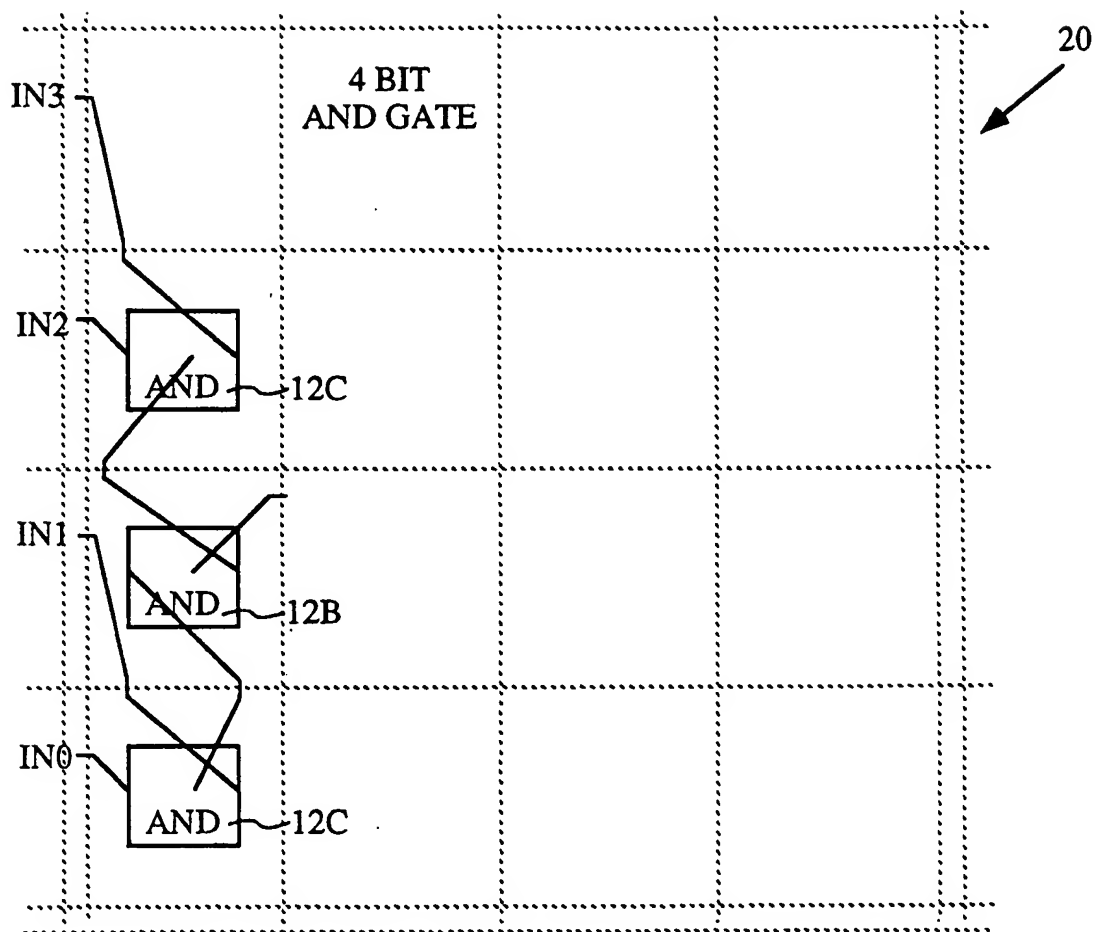


FIG. 40

34/42

**FIG. 40**

35/42

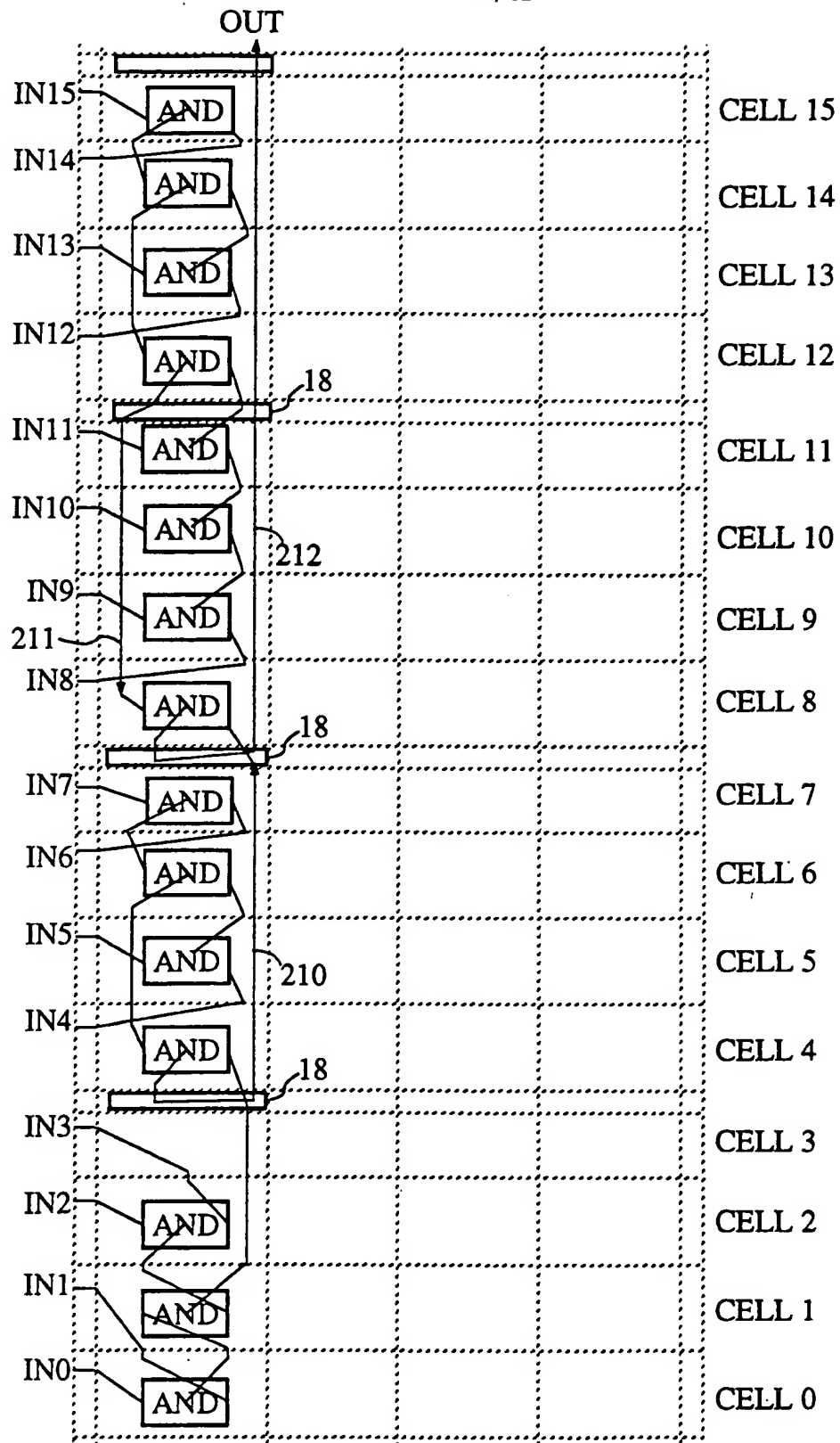


FIG. 41

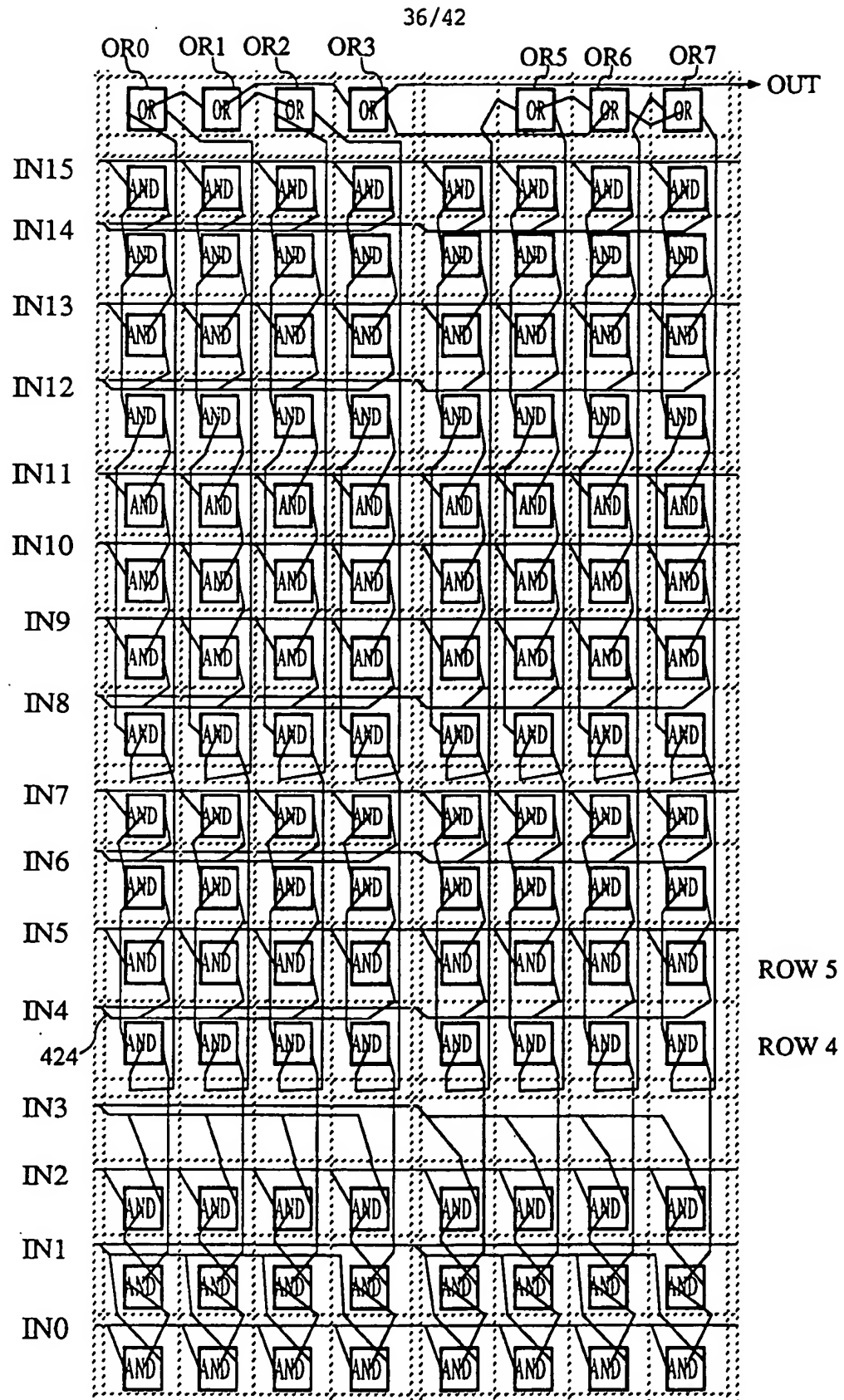


FIG. 42

37/42

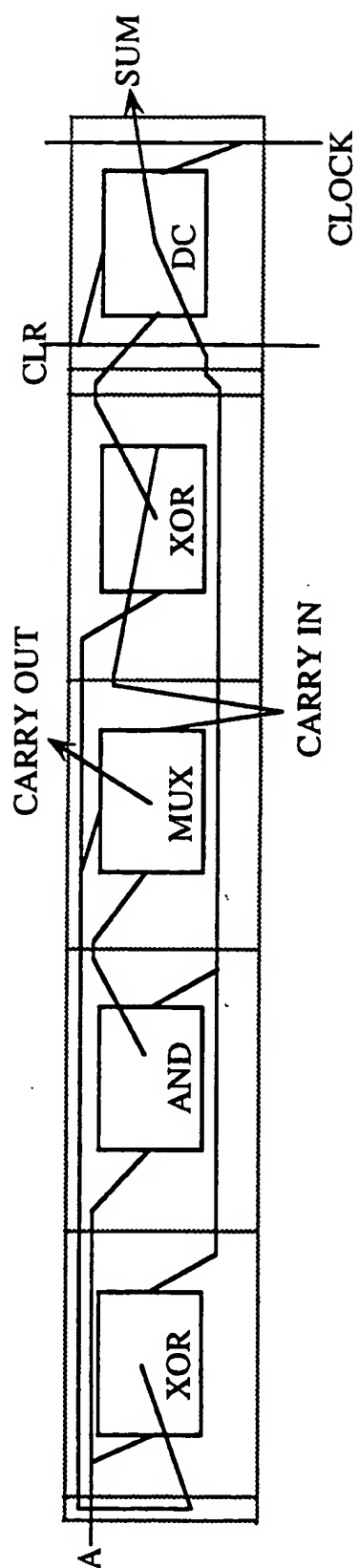


FIG. 43

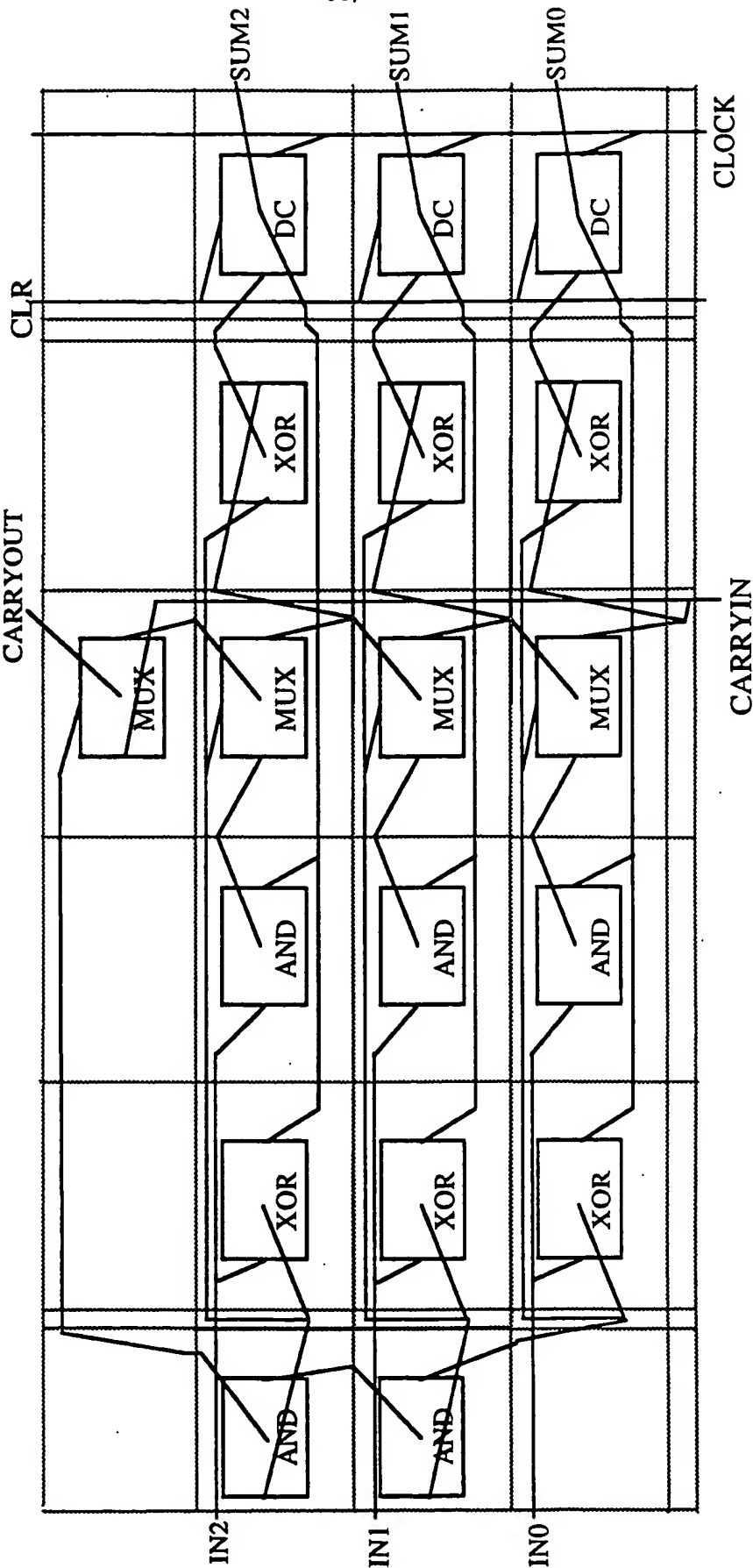
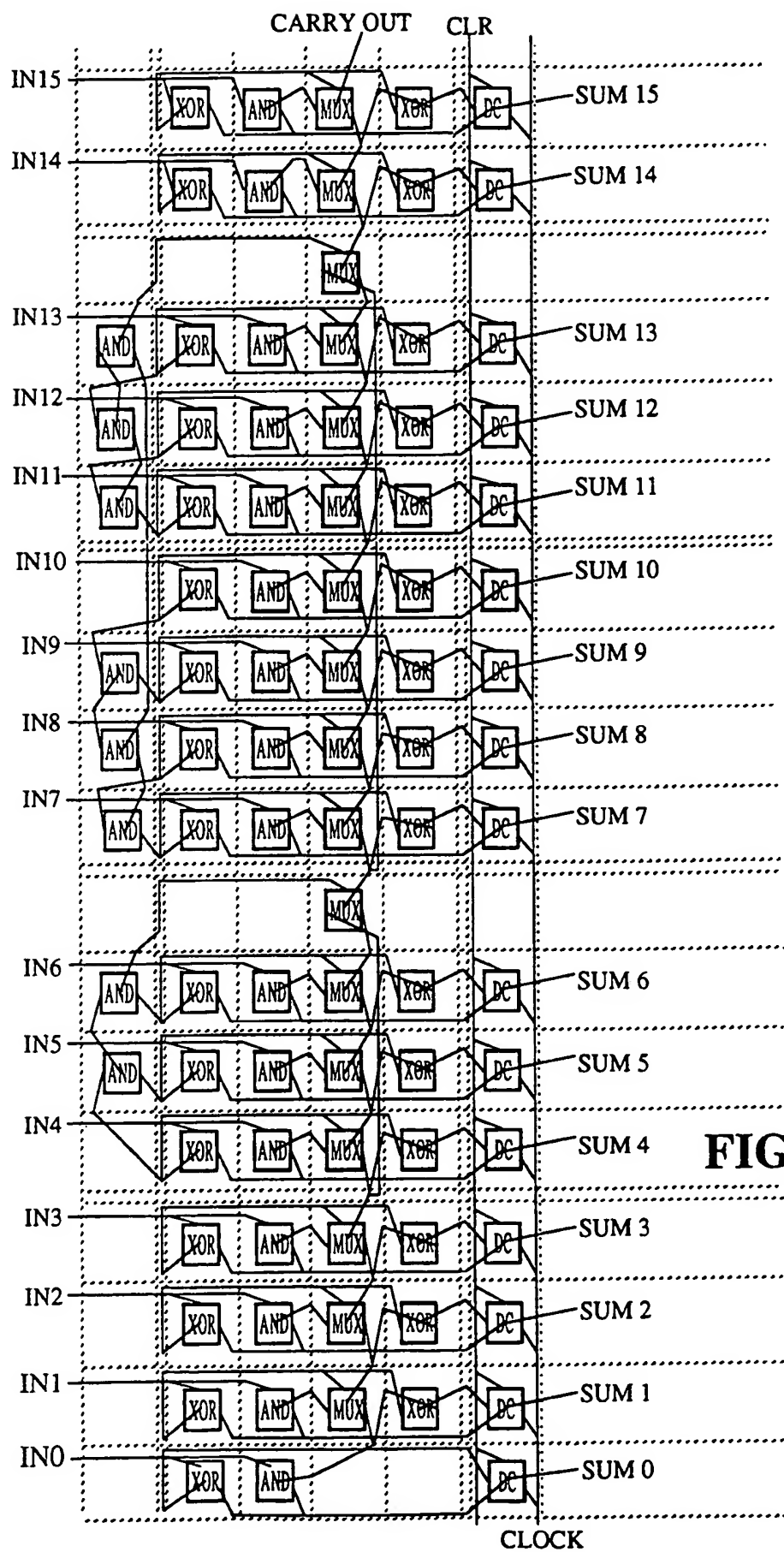


FIG. 44



40/42

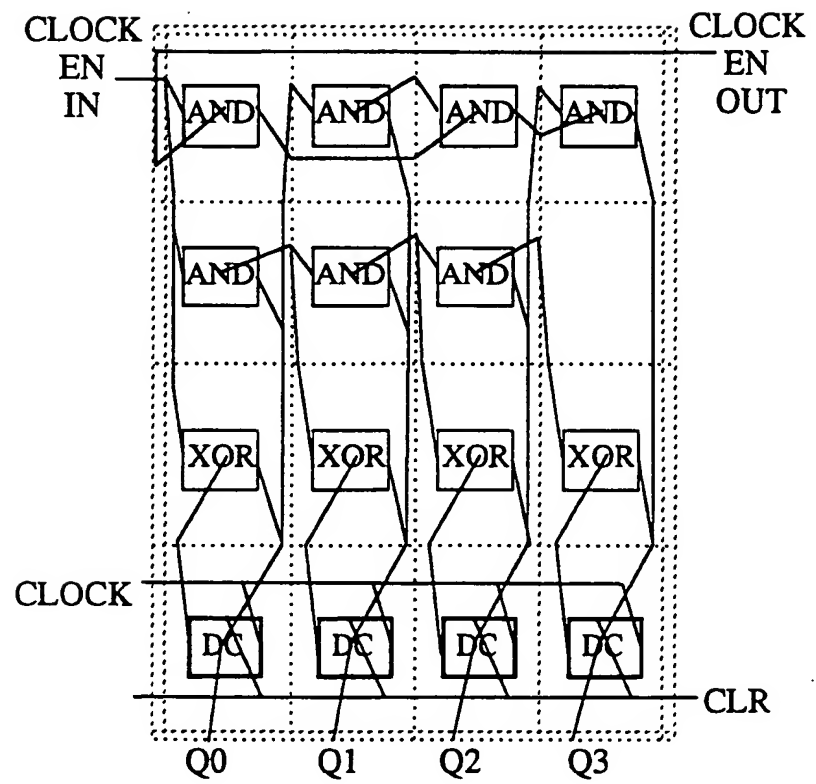


FIG. 46

41/42

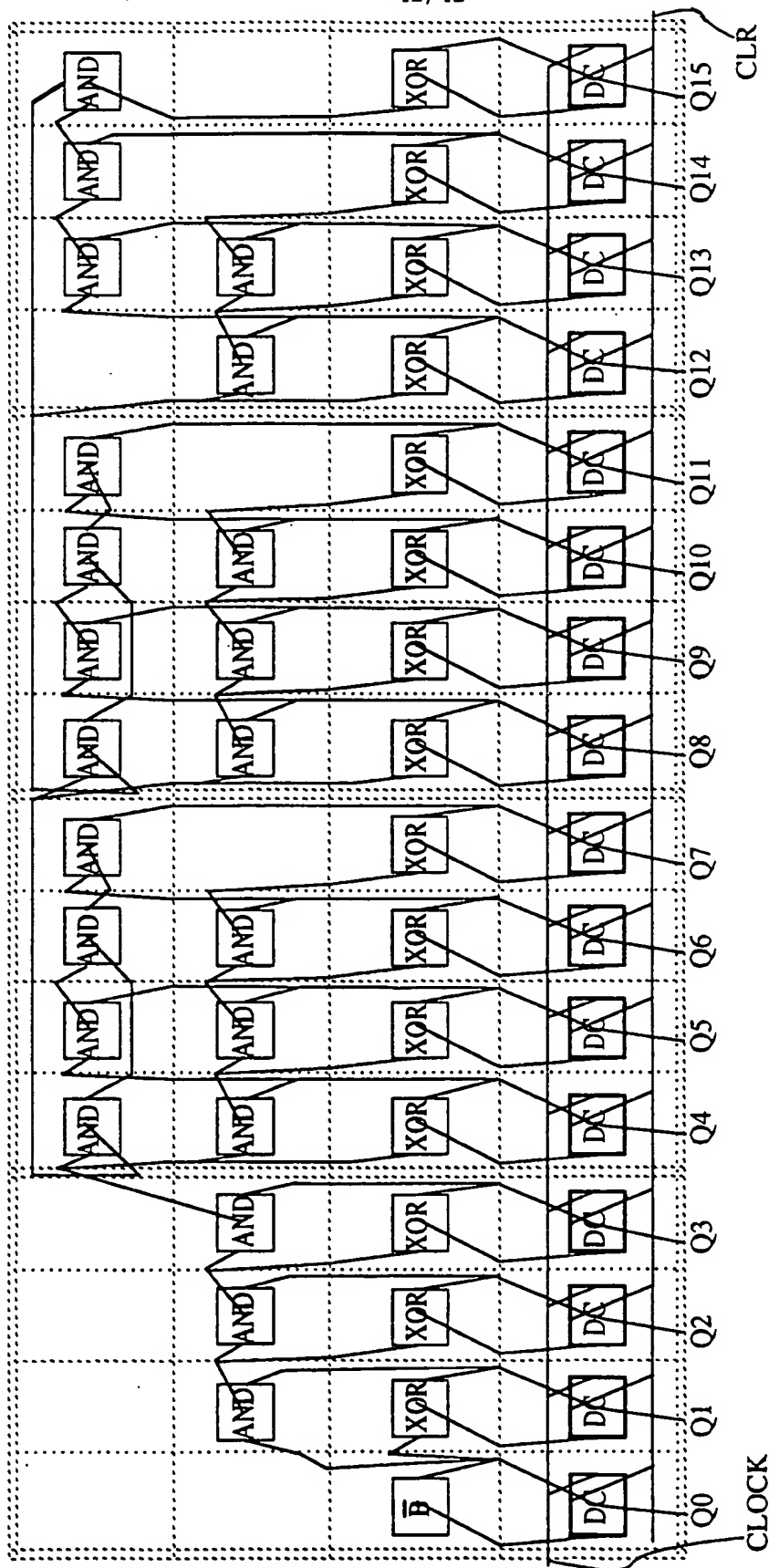
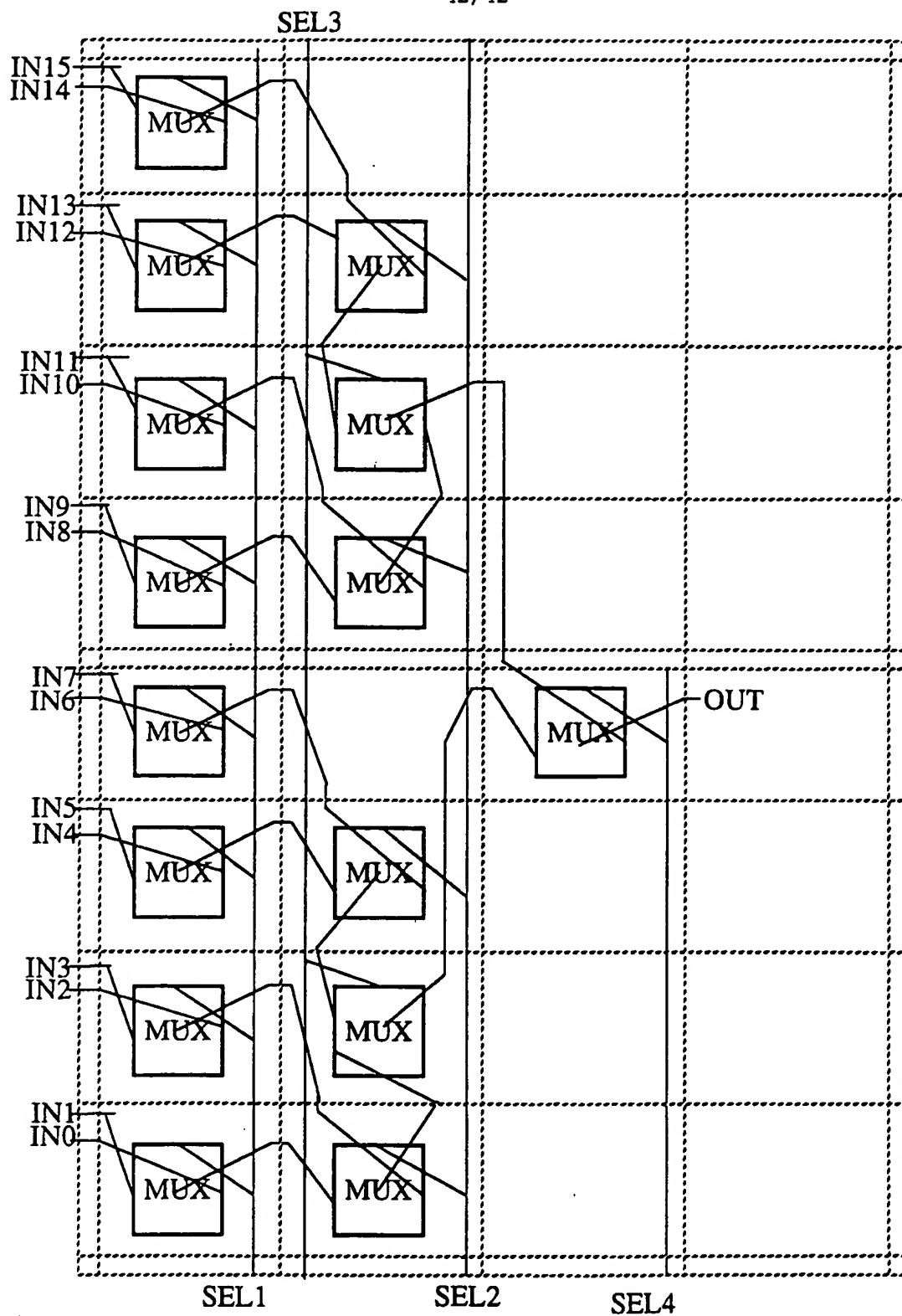


FIG. 47

42/42

**FIG. 48**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/10404

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H03K 19/177

US CL :307/465.1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 307/465.1, 243, 272.2, 441, 443, 465, 468, 475, 303.2; 340/825.83, 825.84, 825.85, 825.86, 825.87, 825.88, 825.89, 825.9, 825.91; 364/490, 491; 365/189.07, 189.08, 230.08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS search terms: wildcard, register, latch, programmable, logic, configuration, memory, switches, pattern.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 5,144,166 (Camarota et al) 01 September 1992, Figs. 3, 11 and col. 3, fifth and sixth paragraphs.	7-8, 22-26
Y,P	US, A, 5,187,393 (El Gamal et al) 16 February 1993, Figs. 4-5, col.7, first and second paragraphs.	1-6
Y	US, A, 5,073,729 (Greene et al) 17 December 1991, Fig. 1.	1-10, 13, 16-18, 23-26
Y	US, A, 5,003,200 (Sakamoto) 26 March 1991, Fig. 1.	19
Y	Mead and Conway, "Introduction to VLSI Systems", October 1980, Addison-Wesley Pub. Co., Reading, Mass., pages 157-160.	33, 36, 40

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"A"	document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means		
"P" document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 16 February 1994	Date of mailing of the international search report 17 FEB 1994
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer for DAVID R. HUDSPETH <i>Dr Miller</i> Telephone No. (703) 308-0956

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/10404

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 4,973,956 (Lin et al) 27 November 1990, Fig. 1.	40
Y	US, A, 4,930,107 (Chan et al) 29 May 1990, Fig. 4A.	30-32
A	US, A, 5,047,983 (Iwai et al) 10 September 1991, Fig. 2.	28-29, 39.
A	US, A, 5,015,883 (Waller) 14 May 1991, Fig. 2.	42.
A	US, A, 4,706,216 (Carter) 10 November 1987, Fig. 4A.	1-26

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/10404

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Telephone Practice

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐

The additional search fees were accompanied by the applicant's protest.

☐

No protest accompanied the payment of additional search fees.

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING

This ISA found multiple inventions as follows:

- I. Claims 1-22 and 41, drawn to a cellular logic array, classified in Class 307, Subclass 465.
- II. Claims 23-27, drawn to a method of routing signals in a programmable logic device, classified in Class 364, Subclass 490.
- III. Claims 28-29, drawn to an apparatus and method of writing to an array of memory cells, classified in Class 365, Subclass 189.08.
- IV. Claims 30-32, drawn to an array of cells with address comparison in a match register, classified in Class 365, Subclass 189.07.
- V. Claims 33-38 and 40, drawn to a selective switch device, classified in Class 340, Subclass 825.88.
- VI. Claim 39, drawn to a decoder having plural duplicate decoders, classified in Class 307, Subclass 441.
- VII. Claim 42, drawn to a function unit comprising multiplexers, classified in Class 307, Subclass 243.

The above groups of claims lack unity of invention under PCT Rule 13.1 - 13.4 because they are related to more than one invention or group of inventions, ^{not} so linked as to form a single general inventive concept. In particular, the method of group II may be used with any matrix of signal processors, not just a cellular logic array (CLA) such as that of group I; the apparatus and method of writing to an array of memory cells in group III may be used in an EEPROM instead of a CLA; the match registers for an array of cells in group IV may be used in a memory device rather than a CLA; the selective switch device of group V may be used in cross-point or barrel switches for a microprocessor instead of a CLA; the redundant decoders of group VI may be used in a memory rather than a CLA; and the function unit comprising multiplexers of group VII may be used as output macrocells in a PAL or PLA, not just in a CLA.